Titre: Title:	Fusion d'analyseurs syntaxiques pour la production d'une analyse syntaxique robuste
Auteur: Author:	Paul Gédéon
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Gédéon, P. (2011). Fusion d'analyseurs syntaxiques pour la production d'une analyse syntaxique robuste [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/654/

UNIVERSITÉ DE MONTRÉAL

FUSION D'ANALYSEURS SYNTAXIQUES POUR LA PRODUCTION D'UNE ANALYSE SYNTAXIQUE ROBUSTE

PAUL GÉDÉON
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

FUSION D'ANALYSEURS SYNTAXIQUES POUR LA PRODUCTION D'UNE ANALYSE SYNTAXIQUE ROBUSTE

présenté par : GÉDÉON, Paul
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

Mme. BELLAÏCHE, Martine, Ph.D., présidente
M. GAGNON, Michel, Ph.D., membre et directeur de recherche
M. OZELL, Benoît, Ph.D., membre et codirecteur de recherche
M. LANGLAIS, Philippe, Doctorat, membre

REMERCIEMENTS

J'aimerais remercier mon superviseur, Michel Gagnon, pour son encadrement et sa patience en plus de ses conseils judicieux basés sur son expertise scientifique. Mon co-superviseur, Benoît Ozell, m'a aussi fourni plusieurs conseils pertinents durant tout le processus de recherche et de rédaction et j'en suis reconnaissant. Ensuite, j'aimerais remercier l'associé de recherche, Éric Charton, pour sa contribution tout autant au niveau scientifique que technique tout au long de mes travaux. Je remercie aussi mes collègues de bureau Jonathan Tardif, François-Xavier Desmarais et Guillaume Bouilly pour leur présence et conseils durant mes expérimentations. Finalement, je voudrais exprimer ma gratitude à mes proches pour leur support et d'avoir été à mes côtés durant les hauts et bas des dernières années.

RÉSUMÉ

Le projet GITAN cherche à traduire automatiquement du texte en une animation 3D. Pour ce faire, plusieurs outils robustes doivent être combinés afin de former le pipeline de traduction automatique. Un de ces outils nécessaires est l'analyseur syntaxique. Par contre, la performance actuelle des analyseurs syntaxiques connus ne satisfait pas les besoins du projet. Pour contrer ce problème, nous cherchons à fusionner les sorties de différents analyseurs pour produire une analyse plus robuste.

Notre principal objectif est d'obtenir une performance maximale et non biaisée à partir d'une fusion d'analyseurs. Pour répondre à cet objectif, nous cherchons à identifier une technique de fusion satisfaisante et une combinaison d'analyseurs qui produit le résultat le plus intéressant.

Trois variantes d'une même technique de fusion principale, soit la technique de vote, sont implémentées. La technique de vote pondérée par catégorie syntaxique a été identifiée comme la plus performante des trois. Cette technique pondère les sorties des analyseurs en fonction de la catégorie syntaxique du mot analysé. Ces pondérations proviennent d'un entraînement sur corpus.

Notre analyseur par fusion produit rarement des arbres syntaxiques invalides. Quand cette anomalie est détectée, nous reconstruisons un arbre syntaxique valide grâce à des algorithmes de découverte d'arbre de recouvrement minimal sur des graphes orientés.

Nous avons ensuite ciblé les analyseurs potentiels pour constituer la combinaison d'analyseurs. Avec la technique de fusion développée et la combinaison d'analyseurs identifiée, nous obtenons des gains de performance intéressants par rapport à la performance du meilleur analyseur faisant partie de la combinaison, atteignant des performances d'environ 91 à 93% de précision.

ABSTRACT

The GITAN project aims to translate automatically text to 3D animation. To do so, many robust tools must be combined to form an automatic translation pipeline. One of those necessary tools is a syntactic parser. However, the actual performance of known parsers doesn't satisfy the needs of our project. To counter this problem, we propose to merge the outputs of different parsers to generate a more robust analysis.

Our main objective is to obtain a maximal and unbiased performance based on parser merging. To achieve this objective, we need to identify a satisfying merging technique and then find the parser combination producing the most interesting result.

Three variants of the same principal merging technique, the voting technique, have been implemented. The voting technique with weights based on part-of-speech is identified as the most interesting of the three variants. This technique adds weights to the parsers output based on the part-of-speech of the word analyzed. Those weights come from a training on a corpus.

Our approach rarely produces invalid syntactic trees. When this anomaly is detected, we reconstruct a valid syntactic tree by applying algorithms discovering minimal spanning trees in directed graphs.

We then identified the potential parsers which will be part of the parser combination. With the developed merging technique and the identified parser combination, we obtained interesting gains of performance in function of the performance of the best individual parser, obtaining performances up from 91 to 93% of precision.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

LAS	Pointage d'attachement libellé (*Labeled Attachment Score*)
UAS	Pointage d'attachement non-libellé (*Unlabeled Attachment Score*)
POS	Catégorie syntaxique (*Part-of-speech*)
CFG	grammaire hors-contexte (*Context-free grammar*)
PCFG	grammaire hors-contexte probabiliste (*Probabilistic context-free grammar*)
MST	arbre de recouvrement minimal (*Minimal spanning tree*)
TAL	Traitement automatique du langage
WSJ	Wall Street Journal

CHAPITRE 1

INTRODUCTION

Dans les dernières années, beaucoup d'efforts ont été consacrés à rendre les différentes branches du traitement automatique du langage (TAL) plus robustes, surtout grâce aux techniques statistiques et aux corpus. Ces performances peuvent être satisfaisantes, mais pas dans tous les contextes.

Effectivement, le projet de recherche dans lequel s'inscrit ce projet de maîtrise consiste à générer une animation automatiquement à partir d'un texte (Ruhlmann *et al.*, 2010). Un tel projet requiert la combinaison de plusieurs outils, tels que des analyseurs syntaxiques et sémantiques ainsi que des outils d'infographie.

Nous pouvons voir ceci comme une traduction automatisée d'une langue naturelle vers une langue graphique. Un projet d'une aussi grande envergure se décompose en plusieurs sous-problèmes qui doivent individuellement offrir une robustesse très élevée. Nous nous attaquons particulièrement à la première étape de la traduction, soit l'analyse syntaxique. Pour mieux se situer dans le projet global, voici une figure permettant de comprendre l'ampleur de la tâche.

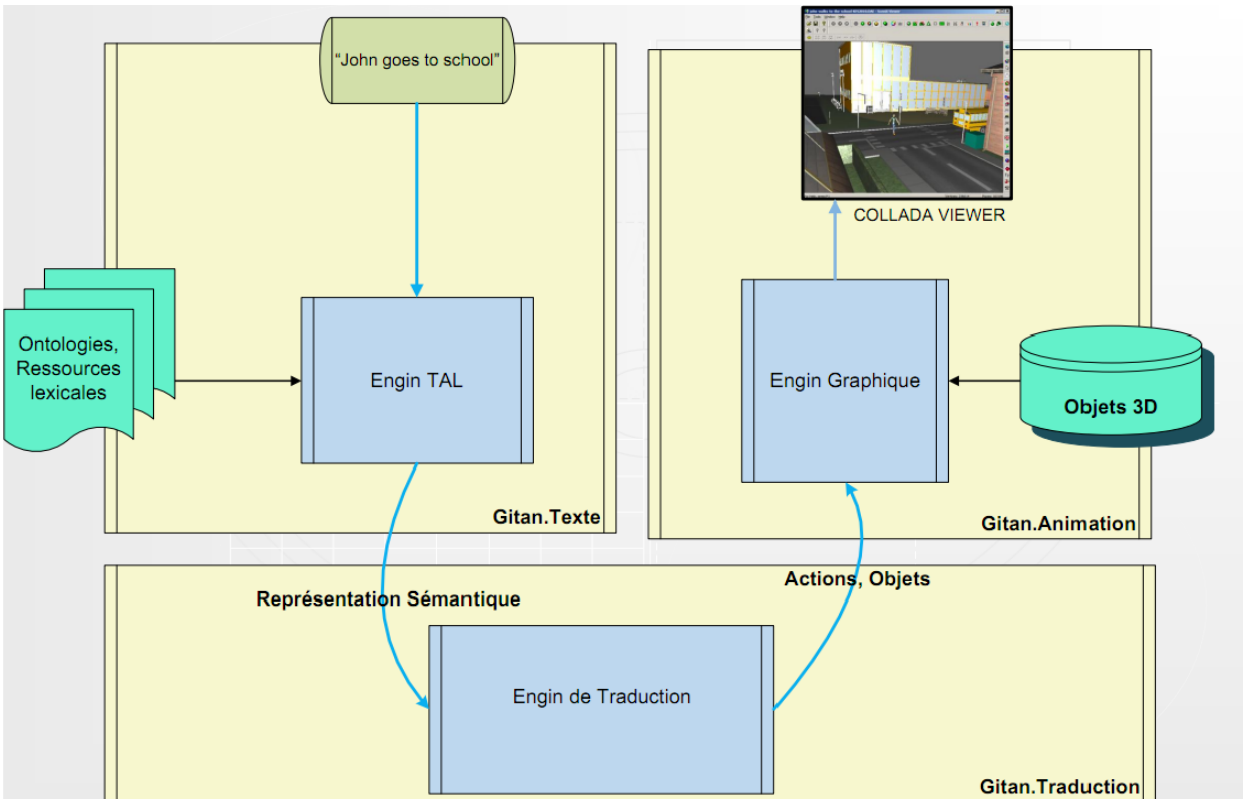

Figure 1.1 Projet global de GITAN

Tout d'abord, la section du projet consacrée au traitement de texte, l'engin TAL, constitue le premier morceau du pipeline. L'analyse syntaxique est la première étape d'un long processus de traitement. Un grand effort est aussi consacré à générer une analyse sémantique plus performante.

Ensuite, il faut développer une passerelle entre le texte et l'infographie, ce que nous appelons l'engin de traduction. Grâce aux ontologies et à la compréhension sémantique du texte, nous développons un langage intermédiaire permettant de bien conceptualiser une scène d'animation.

À partir de ce langage intermédiaire, nous voulons pouvoir générer les objets ainsi que leurs contraintes spatiales. Nous définissons aussi les actions qui font partie de la scène. Une fois que la scène est prête à être animée, nous la générons à partir de la banque de modèles 3D et nous créons les interactions entre les objets. Tout ce traitement se fait dans l'engin graphique.

Ainsi, en comparaison avec le pipeline classique de traduction automatisée, notre pipeline repose sur un nombre beaucoup plus grand de modules. Or, plus nous utilisons d'outils, plus ils doivent être indépendamment robustes pour assurer une propagation d'erreurs minimale. Donc, dans la décomposition du projet général de grande envergure, nous prévoyons d'améliorer la performance de chacun de ces outils. Ce projet de recherche consiste particulièrement à améliorer les performances de l'analyse syntaxique.

Les analyseurs syntaxiques modernes commettent des erreurs environ 3 fois sur 20 sur le corpus de référence (Penn Treebank). Ce type de performance n'est pas assez robuste dans un contexte combinant plusieurs outils comme le projet décrit. Dû à la recherche très poussée déjà effectuée dans le domaine des analyseurs syntaxiques, nous croyons difficile d'augmenter la performance individuelle d'un analyseur. Nous cherchons donc à obtenir un gain de performance par d'autres moyens. Plus particulièrement, nous cherchons à prouver que ces performances peuvent être améliorées afin de rendre l'analyse syntaxique injectée dans le pipeline plus robuste. Pour ce faire, nous avons exploré la technique qui consiste à combiner le résultat de plusieurs analyseurs syntaxiques. Nos résultats montrent que cette technique affiche un potentiel pour augmenter les performances de l'analyse syntaxique.

Dans le Chapitre 1, la problématique est présentée, incluant les hypothèses et les objectifs. Le Chapitre 2 explique en détail les concepts linguistiques dont la compréhension est nécessaire pour le reste de cet ouvrage.

Le Chapitre 3 présente les différentes façons d'effectuer une analyse syntaxique. Ce chapitre inclut aussi une comparaison entre ces méthodes et les métriques pour mesurer leurs performances. La revue de littérature couvrant la fusion d'analyseurs se retrouve dans le Chapitre 4.

L'approche que nous avons élaborée est exposée dans le Chapitre 5. La méthodologie pratique ainsi que le plan d'expérience pour appliquer cette approche théorique sont ensuite présentés exhaustivement dans le Chapitre 6. Le Chapitre 7 contient les résultats de cette démarche et l'analyse de ces données. Finalement, le dernier chapitre, le Chapitre 8, présente les travaux futurs pouvant être effectués à partir de la base expérimentale établie dans nos travaux.

CHAPITRE 2

PROBLÉMATIQUE

2.1 Évolution du domaine

La traduction automatisée classique, qui consiste à transformer un texte en entrée en une sortie représentée dans une autre langue, a fasciné les informaticiens depuis des décennies. La première trace concrète de l'intérêt scientifique envers la traduction automatisée remonte aux années 1940, où Warren Weaver considère les possibilités et limitations de la traduction automatisée dans son mémorandum intitulé "Translation" (Wikipedia, 2011c).

Cette possibilité a piqué la curiosité de plusieurs chercheurs et une première conférence a été organisée sur le sujet dans les années 1950 (Hutchins, 1992). Celle-ci a vite été suivie par quelques projets ambitieux concrets servant de preuves de faisabilité (Hutchins, 2004). Le domaine commençait à prendre de son élan et plusieurs grandes universités se sont mises à créer des groupes de recherche sur le sujet (Hutchins, 1999).

Dans les années 1960, cette vague de curiosité et d'appétit pour un domaine innovateur a vite été refroidie par des rapports officiels qui ont eu pour effet de couper massivement les subventions injectées dans le domaine (Hutchins, 2003). Ce coup de masse a épargné certains groupes de recherche qui ont continué leurs travaux sur des projets innovateurs de toute sorte, comme des systèmes de questions-réponses automatisés (Woods, 1968), un traducteur de l'anglais vers le français de rapports météos (Chevalier *et al.*, 1978) et plusieurs autres.

Il a fallu attendre la fin des années 1970 pour avoir droit à un changement majeur dans le domaine. Les scientifiques de l'époque commençaient à s'intéresser à la traduction automatisée basée sur les transferts. Au lieu d'essayer de traduire le mieux possible le texte original, on essaye désormais de comprendre en profondeur le sens de la phrase avant toute traduction. On cherche à décortiquer la morphologie des mots, trouver leurs racines, faire une analyse syntaxique, parfois même une analyse sémantique et ainsi de suite. Avec une représentation de transfert beaucoup plus riche en information, on peut à présent générer une traduction plus précise (Wikipedia, 2011b).

Durant les années 90, le domaine est devenu encore plus robuste grâce à la révolution des

techniques statistiques (Brown *et al.*, 1990). Effectivement, grâce à l'apparition de corpus rédigés manuellement (Hutchins, 1999), on peut enfin entraîner statistiquement les traducteurs à utiliser des méthodes probabilistes lorsque confrontés à une décision non déterministe. Cette nouvelle façon de procéder a donné naissance à des logiciels encore plus robustes, comme le très connu logiciel en-ligne gratuit Google Translate (Wikipedia, 2011a). De nos jours, on commence à être satisfaits des résultats de la traduction automatisée, autant à l'écrit que pour la parole.

Toutefois, il y a toujours place à la création de nouvelles branches novatrices au domaine. L'une de ces branches concerne la multimodalité, où la traduction ne vise pas le transfert d'un texte vers un autre texte, mais plutôt deux médiums distincts, par exemple un texte et une animation, comme c'est le cas dans le projet GITAN.

2.2 Situation actuelle

Dans le cadre du projet GITAN, nous avons besoin d'une analyse syntaxique plus performante que ce qui est disponible actuellement dans le monde des logiciels libres. Le domaine de l'analyse syntaxique est encore d'actualité dans le domaine de la recherche. Plusieurs sous-thématiques entourant ce sujet principal ont été à l'étude dans diverses campagnes d'évaluation au cours des dernières années (Buchholz et Marsi, 2006) (Nivre *et al.*, 2007a) (Surdeanu *et al.*, 2008).

Néanmoins, les campagnes d'évaluation récentes traitent plutôt de sujets sous-jacents, comme l'évaluation d'analyseurs syntaxiques multilingues (Nivre *et al.*, 2007a) ou bien l'analyse conjointe de syntaxe et de sémantique (Surdeanu *et al.*, 2008).

De plus, les scientifiques s'intéressent plus à l'évaluation de la performance des analyses théoriques qu'à la performance pratique d'un produit fini. Toutefois, une vaste quantité d'outils ont été mis en place pour arriver à un résultat encore plus intéressant. Des techniques de fusion ont été testées et évaluées, mais seulement sur des analyseurs qui ne sont pas parmi les plus performants. Nous cherchons donc à réutiliser ces techniques, mais sur des analyseurs performants pour maximiser le gain de performance.

2.3 Motivation

Dans nos expérimentations, nous ne cherchons pas nécessairement à trouver une nouvelle méthode algorithmique ou conceptuelle, mais plutôt à tester des techniques étudiées sur des analyseurs performants normalisés selon un format connu. Dans le passé, l'approche adoptée consistait à tester et explorer la performance de la fusion en utilisant plusieurs variantes d'un analyseur connu ou de prendre des analyseurs compatibles sans trop se soucier d'utiliser les analyseurs les plus performants. Les expérimentations étaient uniquement centrées sur le gain net de performance.

Dans notre cadre expérimental, le gain de performance optimal est un des objectifs principaux. Par contre, l'obtention d'une performance maximale, peu importe le gain, est aussi primordiale. C'est pourquoi un grand effort a été consacré à rendre compatibles plusieurs des meilleurs analyseurs disponibles actuellement selon un format connu, soit le format de la campagne CoNLL. De la sorte, nous bâtissons un banc d'essai flexible et facilement extensible pour essayer d'insérer d'autres analyseurs connus ou étendre nos recherches avec de nouvelles méthodes de fusion dans le futur. Aussi, étant donné que nous suivons un format connu, nous pourrons insérer des analyseurs plus performants qui pourraient apparaître dans le futur. Nous voulons donc obtenir le maximum de fiabilité possible pour un analyseur syntaxique via l'utilisation de techniques connues, mais dans un cadre expérimental permettant l'obtention d'un maximum de performance.

2.4 Objectifs et hypothèses

Notre hypothèse principale est qu'il est possible d'améliorer la performance des analyseurs syntaxiques actuels grâce aux techniques de fusion. La fusion consiste à prendre plusieurs analyses en entrée et à les comparer afin de générer une seule sortie améliorée. Nous décomposons cette hypothèse de la façon suivante :

- **H1** Les techniques de fusion permettent d'obtenir un gain de performance.
- **H2** Les arbres syntaxiques produits par cette fusion sont des arbres valides.
- **H3** Une combinaison d'analyseurs variés algorithmiquement entraîne un gain de performance supérieur à celui obtenu avec une combinaison d'analyseurs plus performants, mais de même base algorithmique.

- **H4** Une performance maximale supérieure à 90% est atteignable.

Pour trouver des réponses à ces hypothèses, nous nous sommes fixés quelques objectifs. Évidemment, notre objectif principal est d'obtenir une analyse syntaxique la plus précise possible. Notre démarche se décompose de la manière suivante :

- **O1** Trouver les meilleurs analyseurs actuels.
- **O2** Identifier un format pour uniformiser les analyseurs.
- **O3** Choisir une métrique pour mesurer la performance des analyses.
- **O4** Trouver un jeu de données pour effectuer le calcul des pondérations, pour mesurer la performance et pour valider nos conclusions.
- **O5** Tester plusieurs techniques de fusion et déterminer la meilleure.
- **O6** Tester plusieurs techniques de reconstruction et déterminer la meilleure.
- **O7** Tester différentes combinaisons d'analyseurs et déterminer la meilleure.
- **O8** Étudier les effets de la variété algorithmique des analyseurs.

Pour réaliser les objectifs O5 et O6, nous avons effectué plusieurs tests et si une technique est la plus performante dans plus du 2/3 des expériences, nous la conserverons comme meilleure technique.

Pour O7, nous avons adopté une démarche de validation croisée, présentée au Chapitre 6. En résumé, si une combinaison d'analyseurs est identifiée comme meilleure combinaison possible et que durant la validation elle est encore parmi les trois meilleures combinaisons, alors nous considérerons la validation comme un succès.

Pour l'objectif O8, nous comparons la variété algorithmique des analyseurs présents dans les meilleures combinaisons identifiées durant nos expériences. Si notre hypothèse est vraie, alors on s'attend à retrouver dans la combinaison des analyseurs qui ne sont pas aussi performants que d'autres qui ont été ignorés. Ces analyseurs moins performants entraîneraient une plus grande variété algorithmique au sein du comité, ce qui expliquerait leur présence au détriment d'analyseurs plus performants.

CHAPITRE 3

CONCEPTS LINGUISTIQUES

Dans ce chapitre, nous présentons des principes fondamentaux de la linguistique qui sont sous-jacents à notre démarche. Pour pouvoir explorer les techniques d'analyse syntaxique, il faut être bien familier avec ces concepts linguistiques. Nous présentons donc ce qu'est un corpus et les différentes façons formelles de représenter l'analyse syntaxique.

3.1 Corpus

Un corpus est un ensemble de textes annotés validé manuellement. Les types d'annotations contenus dans différents corpus sont variables. Certains corpus contiennent uniquement des informations de base telle que le libellé de la catégorie syntaxique d'un mot ou bien son lemme (sa forme non fléchie). D'autres vont jusqu'au point d'annoter les relations syntaxiques, voire même sémantiques, des phrases. Des corpus se retrouvent sous différents formats de représentation. Des corpus pour différents domaines existent aussi, comme le corpus GENIA dans le domaine biomédical (Kim *et al.*, 2003).

Les corpus sont extrêmement importants durant les évaluations de performance des différentes techniques d'analyse. Effectivement, un jeu de données de plusieurs milliers de phrases permet de mesurer et de comparer les différents analyseurs entre eux. Quand nous pouvons quantifier les résultats d'un analyseur, il est beaucoup plus simple de découvrir comment l'améliorer et de se comparer à un standard.

Un corpus sert de base de données contenant des informations linguistiques à partir desquelles nous effectuons des études statistiques afin de confirmer des hypothèses linguistiques théoriques. Les corpus sont souvent divisés en différentes sections pour les différentes phases d'expérimentation.

Nous voulons utiliser des corpus de campagnes d'évaluations connues comme outil de référence pour comparer les différents analyseurs et ensuite tester les performances des méthodes de fusion. De plus, l'utilisation d'un corpus connu sera critique pour pouvoir situer notre analyseur par rapport à l'état de l'art. Nous nous intéressons ici particulièrement au corpus

du Wall Street Journal, provenant du Penn Treebank (Marcus *et al.*, 1993).

Ce corpus est divisé en trois morceaux : la partie d'entraînement, la partie de développement et la partie de tests. L'entraînement automatique de l'analyseur est effectué sur la section d'entraînement. En utilisant divers classificateurs et techniques d'apprentissages machine, nous extrayons statistiquement du corpus certaines règles que l'analyseur pourra appliquer lors d'étapes non-déterministes de son analyse. Cet ensemble de règles constitue un modèle.

En d'autres mots, les analyseurs sont basés sur des algorithmes qui, à certains moments de leur exécution, doivent prendre une décision où plusieurs possibilités sont valables. Afin de faire le meilleur choix, nous nous référons aux décisions faites par des experts de la linguistique lors de la rédaction manuelle du corpus. Des règles et des patrons sont extraits de ce corpus rédigé manuellement afin de savoir reconnaître la meilleure solution possible à prendre quand l'analyseur est à une étape où il ne peut déterminer une réponse en particulier.

Ensuite, ces règles extraites du corpus durant l'entraînement sont validées sur la section de développement. Effectivement, nous analysons la version qui n'est pas annotée d'informations syntaxiques de la section de développement grâce au modèle construit durant l'entraînement. Par la suite, nous corrigeons notre analyse en la comparant par rapport à la version annotée de la section de développement et nous ajustons les paramètres de notre modèle en fonction des résultats de cette comparaison. Une fois que les paramètres ont été ajustés et que nous avons obtenu un modèle final, nous évaluons les performances de l'analyseur sur la section de test du corpus annoté.

3.2 Représentation formelle de la syntaxe

Il existe plusieurs formalismes pour représenter l'analyse syntaxique d'un texte. Par contre, la quasi-totalité de la littérature porte sur deux formats de représentation syntaxique, soient la représentation par constituants et la représentation par dépendance.

Les bases de la syntaxe par constituants remontent à l'époque hellénistique. Ensuite, plusieurs linguistes ont repris les bases établies par l'Antiquité pour les reformuler de façon beaucoup plus formelle. Quant à elle, la syntaxe par dépendance a commencé à susciter l'intérêt des informaticiens au courant des dernières décennies, alors qu'auparavant seulement la

syntaxe par constituants était étudiée par les linguistes comme Noam Chomsky. (Covington, 2001)

Dans le domaine des corpus annotés, ces deux grandes familles de représentation sont majoritairement utilisées. Par conséquent, notre sélection de représentation est entre ces deux formats puisque l'utilisation de corpus annotés est fondamentale à nos expérimentations. Dans les sections qui suivent, nous présentons et comparons les deux formalismes afin d'expliquer lequel s'adapte le mieux au travail que nous voulons effectuer.

3.2.1 Analyse syntaxique par constituants

L'analyse syntaxique par constituants divise tout d'abord la phrase en plusieurs groupes de mots appelés syntagmes. Un syntagme est un intermédiaire entre l'ensemble global qu'est la phrase et la division unitaire que sont les mots. Le principe de ce concept est de pouvoir subdiviser logiquement la phrase en groupes de plus en plus petits. Le syntagme est en fait un ensemble de mots, ou de plus petits syntagmes, ayant un rôle linguistique commun dans la phrase.

Il existe plusieurs types de syntagmes, comme les syntagmes nominaux (*Un message secret* lui a été envoyé), les syntagmes verbaux (Le chien *ronge un os*), les syntagmes prépositionnels (Il est revenu *pour la rencontrer*), etc. Étant donné que l'on parle de structure hiérarchisée de groupes de mots, une représentation visuelle sous la forme d'arbre est appropriée pour faciliter la lecture de l'analyse. Dans la Figure 3.1, nous représentons le résultat de l'analyse de la phrase « Le superviseur mange une pomme. »

[P [SN[DET Le] [N superviseur]][SV [V mange][SN[DET une] [N pomme]]]]

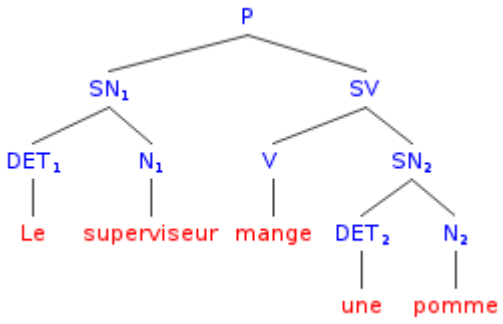

Figure 3.1 Syntaxe de la phrase ”Le superviseur mange une pomme”, représentée par constituants

Par contre, comme nous pouvons le constater, il est difficile de prédire le nombre de syntagmes qui représenteront la phrase. Il est même possible que des analyseurs différents produisent des représentations syntagmatiques de tailles différentes.

3.2.2 Analyse syntaxique par dépendance

Dans l'analyse syntaxique par dépendance, chaque mot est dépendant syntaxiquement d'un autre mot de la phrase, sauf le verbe principal de la phrase qui sera désigné comme la racine de la phrase. Chaque mot, à l'exception de la racine, a exactement une et une seule tête de laquelle il est dépendant. La structure d'arbres peut encore être utilisée pour représenter visuellement l'analyse syntaxique par dépendance.

Les mots de la phrase représentent les nœuds de l'arbre et les relations de dépendance représentent les arcs reliant ces noeuds. Dans la Figure 3.2, nous représentons visuellement la structure par dépendance de la même phrase représentée précédemment, soit « Le superviseur mange une pomme. »

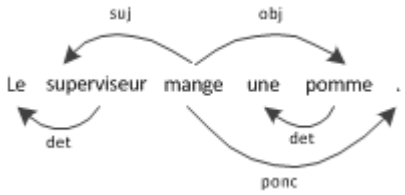

Figure 3.2 Syntaxe de la phrase ”Le superviseur mange une pomme”, représentée par dépendance

Une particularité à noter de ce type d'analyse est la présence d'une racine « artificielle » dans l'arbre. Cette racine est reliée au verbe principal de la phrase et à n'importe quel mot qui n'a pas pu être placé dans l'arbre de dépendance. Cette racine artificielle de l'arbre est nécessaire par technicité pour répondre à des exigences théoriques de certaines techniques utilisées, mais les détails seront omis dans ce résumé.

Une autre propriété des relations de dépendance est la notion de dépendance endocentrique/exocentrique. Dans une relation endocentrique, la tête peut remplacer le mot qui en est dépendant. Par exemple, si nous prenons la phrase « Economic news had little effect on financial markets » de la Figure 3.3, la relation d'attribut (ATT) entre « markets » et « financial » est endocentrique.

Ceci est dû au fait que la phrase « Economic news had little effect on markets » est encore viable. Par contre, la relation de complément prépositionnel entre « on » et « markets » est exocentrique, car « Economic news had little effect on » n'est pas une phrase correcte.

Finalement, notons qu'un arbre de dépendance peut être projectif ou non-projectif. Un arbre de dépendance est projectif lorsqu'il n'y a pas de croisements entre les relations de dépendance au sein de la phrase.

Pour mieux illustrer, deux figures sont respectivement présentées, une projective et l'autre non-projective. Ces deux analyses sont présentées aux Figures 3.3 et 3.4, provenant de l'ouvrage "Dependency Parsing" (Kubler *et al.*, 2009). Nous remarquons que la relation entre les mots *hearing* et *on* croise celle entre *scheduled* et *today*.

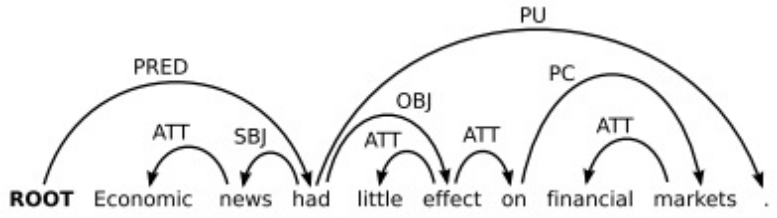

Figure 3.3 Exemple de phrase projective

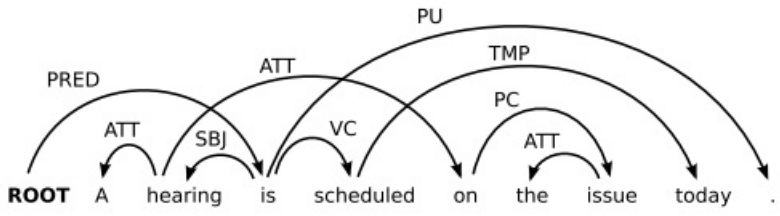

Figure 3.4 Exemple de phrase non-projective

3.2.3 Comparaison entre les représentations

L'analyse syntaxique par dépendance permet un traitement facile par la machine, facilitant l'apprentissage supervisé et l'application d'algorithmes classiques (Kubler *et al.*, 2009). Effectivement, les arbres de dépendance représentent une façon hiérarchisée de structurer l'information où chaque mot est lié à un mot de tête dont il est dépendant.

Contrairement à l'analyse syntaxique par constituants, où le nombre de syntagmes représentant la phrase ne peut être prédit à l'avance, chaque analyse produite contient un nombre fixe d'éléments de représentation. Par conséquent, en sachant que chaque mot n'a qu'une et une seule tête, l'analyse syntaxique par dépendance contiendra exactement un élément de représentation pour chaque mot.

En plus, avec une structure simplifiée de représentation où l'information pertinente consiste seulement en le mot, sa tête et la relation de dépendance, les corpus de données sont faciles à manipuler expérimentalement. Ainsi, nous pouvons facilement extraire de l'information à partir d'un texte et faire un traitement automatisé.

De plus, l'analyse syntaxique par dépendance fonctionne de manière à chercher la tête d'un mot à la fois, analysant ainsi la phrase une unité à la fois. Ceci facilite beaucoup le traitement informatique puisque ce traitement est effectué sur un mot à la fois plutôt que de devoir manipuler tous les morceaux syntagmatiques en même temps (Covington, 2001). Dans ses publications, Abney affirme aussi que décomposer une phrase en relations de dépendance est aussi la façon naturelle de faire par le cerveau humain pour analyser syntaxiquement une phrase.

Par ailleurs, étant donné que l'analyse syntaxique est souvent suivie par une analyse sémantique, nous cherchons à faciliter celle-ci avec une représentation plus appropriée. Or, la représentation par liens de dépendance est beaucoup plus proche des liens sémantiques que

la représentation syntagmatique (Covington, 2001).

CHAPITRE 4

ALGORITHMES D'ANALYSE SYNTAXIQUE

Dans ce chapitre, nous nous concentrons sur les différentes façons d'effectuer une analyse syntaxique. Ensuite, nous présentons les différentes métriques qui servent à mesurer la performance de ces analyses.

4.1 Types d'analyses syntaxiques

Nous étudions ici trois catégories d'analyses syntaxiques : l'analyse syntaxique basée sur les transitions, l'analyse syntaxique basée sur les graphes et l'analyse syntaxique basée sur les grammaires.

4.1.1 Analyse syntaxique basée sur les transitions

Le principe d'analyseur basé sur les transitions est basé sur le « shift-reduce parsing », qui est une technique d'analyse assez connue. Cette technique a besoin de deux conteneurs d'information, une pile et un tampon. Initialement, tous les mots de la phrase sont contenus dans le tampon et la pile est vide. Au cours de l'analyse, la pile contiendra les mots traités.

Trois sortes d'opérations sont possibles : transition arc-gauche, transition arc-droit et translation. Nous ajoutons aussi l'opération de réduction, qui permet d'enlever un mot de la pile si celui-ci a déjà une tête. La translation permet de prendre le mot au début du tampon et de le déplacer vers la pile.

La transition arc-gauche ajoute un arc de dépendance entre la tête du tampon et la tête de la pile, où le mot sur la tête de la pile est le mot-tête de la dépendance. Ensuite, le mot sur la tête de la pile est enlevé.

La transition arc-droit ajoute un arc de dépendance entre la tête du tampon et la tête de la pile, où le mot sur la tête du tampon est le mot-tête de la dépendance. Ensuite, le mot sur la tête du tampon est ajouté sur le dessus de la pile.

L'exemple suivant applique la technique "shift-reduce" pour faciliter la compréhension :

Tableau 4.1 Exemple étape par étape de la technique "shift-reduce" pour la phrase "Le superviseur mange une pomme."

Opération	Pile	Tampon	Dépendance
Translation	[ROOT, Le]	[superviseur, ...]	-
Arc-gauche	[ROOT]	[superviseur, ...]	(superviseur, DET, Le)
Translation	[ROOT, superviseur]	[mange, ...]	-
Arc-gauche	[ROOT]	[mange, ...]	(mange, SBJ, superviseur)
Arc-droit	[ROOT, mange]	[une, ...]	(ROOT, ROOT, mange)
Translation	[ROOT, mange, une]	[pomme, .]	-
Arc-gauche	[ROOT, mange]	[pomme, .]	(pomme, DET, une)
Arc-droit	[ROOT, mange, pomme]	[.]	(mange, PRD, pomme)
Réduction	[ROOT, mange]	[.]	-
Arc-droit	[ROOT, mange, .]	[]	(mange, P, .)

Par contre, pour réussir à déduire quelle opération effectuer dans un état donné, il faut avoir une sorte d'oracle auquel se référer. Il existe plusieurs techniques d'approximation d'oracles comme les grammaires formelles et les heuristiques de désambiguïsation. Par contre, la technique la plus populaire est l'entraînement de classificateurs sur des banques d'arbres.

Tout d'abord, pour pouvoir entraîner des classificateurs pour ce problème, il faut pouvoir représenter les données à traiter d'une façon traitable par la machine. Du coup, nous identifions principalement chaque mot avec son indice de position dans la phrase, son étiquette de catégorie syntaxique (*part-of-speech*) (POS) et sa dépendance dans la phrase.

Le classificateur est entraîné sur les banques d'arbres et les instances découvertes sont répertoriées durant l'entraînement. Ensuite, lorsque nous utilisons l'analyseur, nous trouvons les k instances les plus semblables grâce à des métriques de ressemblances et le résultat est extrapolé selon ces données. Nous avons donc notre approximation d'oracle pour utiliser le « shift-reduce parsing ».

4.1.2 Analyse syntaxique basée sur les graphes

Un des concepts fondamentaux des analyseurs basés sur les graphes est la notion de pointage. En effet, chaque graphe est évalué par un pointage qui indique la viabilité de ce

graphe d'être la solution valide. En fait, ce pointage est la combinaison des pointages des sous-graphes. Par contre, un abus de notation est utilisé ici puisque nous travaillons dans un modèle basé sur les arcs.

De la sorte, les pointages sont attribués aux arcs plutôt qu'au sous-graphe contenant les deux nœuds aux extrémités de l'arc. Le terme "MST" est aussi utilisé pour référer au graphe avec le meilleur pointage. Des libellés sont aussi attachés aux arcs afin d'indiquer la relation de dépendance entre les deux noeuds.

Avant de commencer le traitement, nous devons passer d'analyse libellée à non-libellée. La raison est simple ; nous voulons utiliser des algorithmes fonctionnels et connus, mais ceux-ci sont seulement compatibles avec une structure non-libellée. Il faut donc retirer les libellés des arcs pour analyser la phrase. Toutefois, ces informations sont conservées dans une mappe renversée, qui nous permettra de libeller à nouveau les dépendances une fois le traitement effectué.

Il existe des algorithmes pour l'analyse projective et non-projective. Pour les analyseurs non-projectifs, nous nous basons sur l'algorithme de Chu-Liu-Edmonds. Par fin de vulgarisation, seulement les grandes lignes de cet algorithme seront présentées. Un exemple tiré du livre "Dependency Parsing" (Kubler *et al.*, 2009) est utilisé comme référence pour faciliter la compréhension.

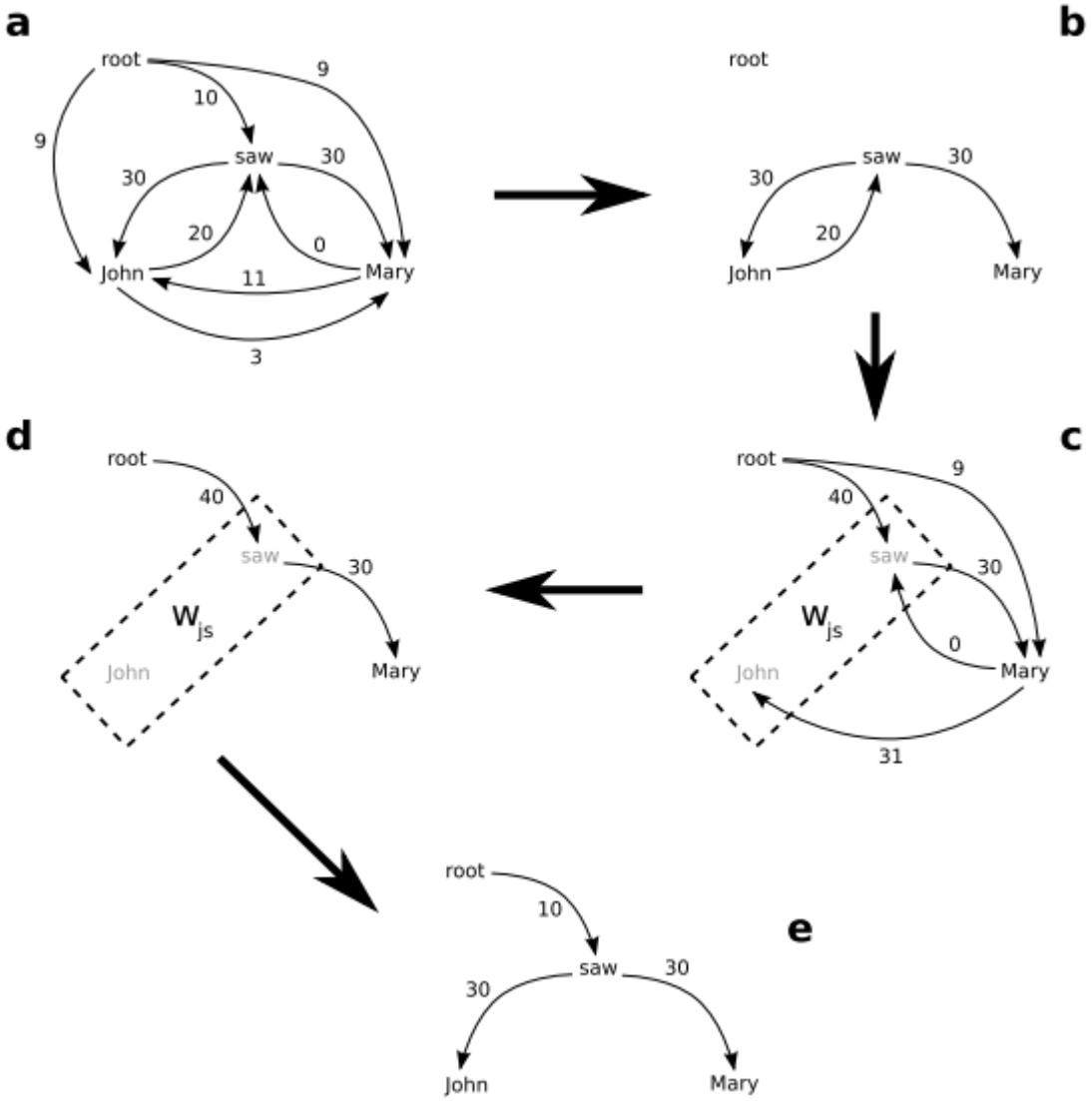

Figure 4.1 Application de l'algorithme d'Edmonds pour la phrase "John saw Mary."

Avant d'appliquer l'algorithme d'Edmonds, il faut fixer les poids initiaux des arcs grâce un modèle construit par apprentissage automatisé. Une fois les poids initiaux fixés, nous pouvons débuter l'analyse. Tout d'abord, nous trouvons l'arc entrant avec la plus grande valeur pour chaque nœud.

Par exemple, à l'étape a, il y a trois arcs entrants pour le noeud *saw*, soit l'arc de poids 10 de *root* à *saw*, l'arc de poids 20 de *John* à *saw* et l'arc de poids 0 de *Mary* à *saw*. L'arc avec le plus gros poids est l'arc de poids 20 de *John* à *saw*. Il sera donc conservé. Ensuite, les autres arcs entrants pour ce nœud sont éliminés, soit l'arc de *root* à *saw* et l'arc de *Mary* à *saw* (étape a à b de la Figure 4.1).

Par la suite, nous trouvons les cycles dans le graphe et nous les regroupons de manière à ce qu'ils soient représentés par un seul noeud par cycle. Dans notre exemple, nous avons gardé l'arc de *John* à *saw* à l'étape précédente puisque c'était l'arc entrant avec le plus haut poids pour le noeud *saw*. Par contre, nous avons aussi gardé l'arc de *saw* vers *John* puisque c'est l'arc entrant avec le plus haut poids pour le noeud *John*. Ces deux arcs forment donc un cycle et nous allons le représenter par le noeud W_{js} (étape b à c de la Figure 4.1).

Nous avons donc un nouveau graphe réduit puisqu'un cycle a été remplacé par un seul noeud. Il est à noter que le poids des arcs qui entraient dans le cycle est remplacé par le poids du plus long chemin amenant dans le cycle. Par exemple, l'arc de *root* vers *saw* de poids 10 est remplacé par un arc de poids 40. Ceci est dû au fait qu'il existe un chemin de *root* vers *John*, un autre membre du cycle, de poids 40. Ce chemin est constitué de l'arc de poids 10 de *root* vers *saw* suivi de l'arc de poids 30 de *saw* vers *John*. Ces deux arcs sont donc combinés pour former un seul arc entrant de poids 40 de *root* vers W_{js}.

Sur ce nouveau graphe, nous appliquons à nouveau la première étape, qui consistait à ne garder que l'arc entrant avec le plus haut poids pour chaque noeud (étape c à d de la Figure 4.1). À l'étape d, nous conservons donc le nouvel arc de poids 40 de *root* vers W_{js} ainsi que le même arc entrant de poids 30 pour le noeud *Mary*.

En se rappelant des points terminaux (*endpoints*) originaux, un arbre peut finalement être construit une fois que l'élimination des cycles a été complétée (étape d à e de la Figure 4.1). Par la suite, l'arbre généré est traduit en arbre de dépendance en ajoutant les relations de dépendance associées à chaque branche.

Pour les analyses projectives, l'algorithme Cocke-Younger-Kasami est appliqué. Le principe de cet algorithme est de calculer toutes les séquences et sous-séquences de mots de taille 1 à n, où n est le nombre de mots dans la phrase. L'idée est de commencer par trouver le pointage de tous les sous-graphes de taille 1, ensuite prendre les sous-graphes adjacents et trouver le pointage de tous ces graphes de taille 2 et ainsi de suite.

Toutefois, cet algorithme prend $O(n^3)$ pour être exécuté, ce qui est un problème significatif et rend l'algorithme beaucoup moins intéressant. Évidemment, des variantes à l'algorithme comme l'algorithme d'Eisner ont été trouvées pour répondre à ce problème de complexité. Cet algorithme est toutefois assez complexe et sa compréhension approfondie n'est pas nécessaire pour la suite de nos travaux, donc nous n'élaborerons pas davantage sur son fonctionnement.

L'approche basée sur les graphes est aussi pilotée par les données. Tout d'abord, il faut représenter les données que nous voulons tirer de l'entraînement. Nous représentons des caractéristiques linguistiques d'un arc comme l'étiquette POS du nœud de l'arc entrant et sortant et des poids sont attribués à ces caractéristiques selon leur importance.

Ensuite, durant l'entraînement sur une banque d'arbres, nous trouvons l'arbre avec le meilleur pointage. Nous prenons subséquemment les caractéristiques présentes dans cet arbre et nous leur ajoutons un poids positif, alors qu'un montant est soustrait au poids pour les caractéristiques non-présentes.

4.1.3 Analyse syntaxique basée sur les grammaires

Les techniques d'analyse basées sur la grammaire utilisent tous les principes fondamentaux d'une grammaire hors-contexte (*context-free grammar*) (CFG). Nous retrouvons donc les notions de symboles terminaux, non-terminaux et de règles de production. En fait, nous pouvons plutôt voir les grammaires de dépendance projective comme un cas spécial des CFG où les symboles terminaux sont des mots.

Plus formellement, nous définissons une grammaire hors-contexte (Kubler *et al.*, 2009) comme un 4-tuple (N, Σ, Π, S) où :

- N est un ensemble de symboles non-terminaux.
- Σ est un ensemble de symboles terminaux.
- S est le symbole de de départ et appartient à l'ensemble N.

– Π est un ensemble de règles de production $X \rightarrow \alpha$, où X est un et un seul symbole appartenant à N et α une chaîne de symboles terminaux et non-terminaux.

Ensuite, une notion statistique est ajoutée à ces concepts, transformant le CFG en grammaire hors-contexte probabiliste (*probabilistic context-free grammar*) (PCFG).

Pour chaque règle de production, une probabilité d'application est ajoutée. Ces probabilités sont calculées par apprentissage (automatisé) sur un corpus. Par exemple, pour deux règles avec le même symbole non-terminal à gauche, des probabilités sont ajoutées aux deux règles pour un total de 1,00. Ainsi, quand nous voulons transformer un symbole non-terminal, la règle à appliquer est choisie de façon probabiliste. L'exemple qui suit a pour but de faciliter la compréhension :

– $N = \{$S, SV, SN, SA, SP$\}$ (où SV est un syntagme verbal, SN un syntagme nominal, SA un syntagme adjectival, SP un syntagme prépositionnel)
– $\Sigma = \{$N, V, Adj, Prep, Det, $\epsilon\}$ (où N est un nom, V est un verbe, Adj est un adjectif, Prep est une préposition, Det est un déterminant et ϵ est un élément vide.
– $S = DEBUT$

L'ensemble de règles de production Π, inspirés des acétates du cours CS388 de University of Texas (Raymond J. Mooney, Année inconnue), est constitué de :

Tableau 4.2 Analyse syntaxique basée sur les grammaires : règles de production Π

Règle	Probabilité
DEBUT $\rightarrow$ SN SV	0,95
DEBUT $\rightarrow$ SV	0,05
SN $\rightarrow$ Det SA N	0,80
SN $\rightarrow$ SN SP	0,20
SA $\rightarrow \epsilon$	0,70
SA $\rightarrow$ Adj SA	0,30
SP $\rightarrow$ Prep SN	1,00
SV $\rightarrow$ V SN	0,70
SV $\rightarrow$ SV SP	0,30

Par exemple, pour générer la phrase "Le brillant superviseur mange une pomme.", l'analyseur doit avoir appliqué les règles de production suivantes :

Tableau 4.3 Analyse syntaxique basée sur les grammaires : construction par grammaire

Règle appliquée	Résultat
DEBUT $\rightarrow$ SN SV	SN SV
SN $\rightarrow$ Det SA N	Det SA N SV
SA $\rightarrow$ Adj SA	Det Adj SA N SV
SV $\rightarrow$ V SN	Det Adj SA N V SN
SN $\rightarrow$ Det SA N	Det Adj SA N V Det SA N
SA $\rightarrow \epsilon$	Det Adj SA N V Det ϵ N
SA $\rightarrow \epsilon$	Det Adj ϵ N V Det ϵ N
-	Le brillant superviseur mange une pomme.

4.1.4 Comparaison entre les types d'analyse syntaxique

Tout d'abord, il faut noter que les analyseurs basés sur les graphes favorisent l'exactitude de l'analyse plutôt que la possibilité d'expression des caractéristiques linguistiques. Bien que ceci améliore les performances, nous perdons beaucoup de possibilités au niveau de l'extraction d'informations.

Les techniques d'analyse basées sur les transitions souffrent de propagation d'erreurs et l'exactitude est donc moindre pour les décisions faites tardivement. De la sorte, les méthodes basées sur les transitions sont moins performantes pour les arcs proches de la racine ou qui ont une plus grande longueur d'arc.

Toutefois, pour les décisions prises tôt en analyse, comme pour les arcs se trouvant plus loin de la racine ou qui ont une plus petite longueur d'arc, il a été noté que les méthodes basées sur les transitions détiennent de meilleurs résultats.

Si nous comparons au niveau du POS, les analyseurs basés sur les transitions ont une meilleure exactitude pour les noms et pronoms, alors que les méthodes basées sur les graphes sont meilleures dans les autres catégories, en particulier pour les conjonctions et verbes. Ceci est consistant avec ce qui a été dit plus tôt, puisque les noms et pronoms sont attachés au verbe et sont souvent plus loin dans l'arbre avec des plus court arcs.

4.2 Métriques et évaluation d'analyseurs

Les méthodes d'évaluation d'analyseurs de dépendance sont basées sur le pointage d'attachement non-libellé (*unlabeled attachment score*) (UAS) et le pointage d'attachement libellé (*labeled attachment score*) (LAS). Le UAS est en fait le nombre de mots associés à la bonne tête dans la phrase. Il y a deux sortes de UAS, soit celui basé sur les mots (UAS macro) et l'autre basé sur les phrases (UAS micro).

Le UAS macro est une moyenne basée sur le taux de succès de chaque mot alors que le UAS micro est une moyenne des taux de succès par phrase. Par exemple, prenons une phrase avec 9 mots corrects sur 10 et une deuxième phrase avec 15 mots corrects sur 45. Formellement, ceci donne :

$$UAS_{macro} = \frac{Total_{correct}}{Total_{mots}} = \frac{9+15}{10+45} = 0,436$$

Le UAS micro calcule le UAS de chaque phrase et fait ensuite la moyenne de tous les UAS de chaque phrase. Le UAS d'une phrase courte aura autant de poids dans la moyenne que le UAS d'une longue phrase. Ce UAS a donc tendance à être plus elevé puisque les phrases courtes sont mieux analysées. Reprenons notre exemple des deux phrases précédentes, mais cette fois pour le UAS de phrase :

$$UAS_1 = \frac{Total_{correct}}{Total_{mots}} = \frac{9}{10} = 0,9$$

$$UAS_2 = \frac{Total_{correct}}{Total_{mots}} = \frac{15}{45} = 0,333$$

$$UAS_{micro} = Moyenne(UAS_1, UAS_2) = \frac{UAS_1 + UAS_2}{2} = 0,617$$

Le LAS est comme le UAS, mais en plus de compter les mots attachés à la bonne tête, nous comptons le nombre de mots attachés à la bonne tête avec la bonne relation de dépendance. Néanmoins, notre métrique pour l'ensemble de notre expérience sera le UAS macro puisqu'il est le plus commun pour évaluer un analyseur par rapport à l'état de l'art.

CHAPITRE 5

FUSION D'ANALYSEURS : L'ÉTAT DE L'ART

Dans ce chapitre, nous présentons les principaux travaux concernant le sujet de fusion d'analyseurs. Cinq principaux articles sont présentés et résumés. De plus, ces articles illustrent bien l'évolution des techniques utilisées, passant de l'idée fondamentale de fusion à des concepts plus élaborés.

5.1 Évaluation d'analyse syntaxique

Lin a été le premier à suggérer l'utilisation de structures par dépendance pour l'évaluation d'analyseurs syntaxiques dans son article (Lin, 1995). La technique qu'il propose consiste à comparer les résultats d'un analyseur à la clé fournie par un oracle.

Anciennement, les analyseurs syntaxiques fonctionnaient avec une structure de constituants, avec des métriques d'évaluations variées et complexes. Désormais, Lin suggère de tout simplement trouver les relations de dépendances erronées. Chaque mot doit nécessairement avoir une tête, donc une relation de dépendance erronée est lorsqu'un mot n'est pas associé à la tête attendue.

Une fois les relations erronées identifiées, il suffit d'en faire le compte. Il nomme cette métrique *error count*. Cette méthode d'évaluation est beaucoup plus intuitive que les anciennes, puisque c'est une proportion directe entre les relations erronées et le nombre total de relations.

Il explique aussi comment modifier les arbres de dépendance avant le traitement pour améliorer les performances. Il cite en particulier certains nœuds, comme des négations et le mot *to* en anglais. La raison est qu'il y a plusieurs façons d'analyser correctement ces mots et qu'ils faussent ainsi grandement les résultats.

Il montre trois opérations qui peuvent être appliquées à des arbres de dépendance, soit l'effacement d'une dépendance, l'inversion et le transfert de dépendance. Il explique que ces opérations peuvent être très utiles pour éliminer des différences inconséquentes qui pourraient nuire aux résultats.

5.2 La diversité dans la combinaison d'analyseurs

Henderson et Brill proposent l'idée novatrice de combiner plusieurs analyseurs syntaxiques basés sur la méthode d'analyse par constituants. Pour ce faire, deux types de techniques de combinaisons sont présentés. La première technique est appelée hybridation d'analyseurs (*parser hybridization*). Cette méthode consiste à parcourir les constituants un après l'autre, et pour chacun décider s'il doit être conservé. Deux sous-méthodes sont ensuite présentées, soit le vote par constituant et la méthode bayésienne naïve.

Le vote par constituant est une méthode assez intuitive. Comme préalable, il nous faut au minimum trois analyseurs pour toujours avoir des votes majoritaires. Ensuite, nous passons un à un les constituants et nous votons afin de savoir quelle est la version de ce constituant qui est majoritairement présente dans les différentes représentations syntaxiques.

Par contre, cette méthode traite tous les analyseurs également, alors que certains doivent nécessairement être meilleurs que les autres ou bien offrir de meilleures performances dans certains contextes.

La méthode bayésienne naïve se base sur l'hypothèse que les erreurs sont distribuées indépendamment dans les différents analyseurs. Nous cherchons à savoir à quel point il faut faire confiance à chaque analyseur dans ses décisions grâce à la classification. Le vote est ensuite pondéré avec les probabilités déterminées par l'apprentissage.

Voici le développement plus formel de la méthode bayésienne tel que défini dans leur article. $\pi(\mathrm{c})$ est une fonction binaire qui retourne *vrai* si le constituant devrait être gardé. $M_k(\mathrm{c})$ est une fonction binaire qui retourne *vrai* si le constituant c devrait être choisi comme tête de la dépendance selon l'analyseur k.

Après quelques transformations appliquées à la probabilité conditionnelle initiale en suivant le théorème de Bayes, nous obtenons l'équation 3.

$$\arg\max_{\pi(c)} \; P(\pi(c))|M_1(c)...M_k(c))$$

$$= \arg\max_{\pi(c)} \quad \frac{P(M_1(c)...M_k(c)|\pi(c))P(\pi(c))}{P(M_1(c)...M_k(c))} \qquad \textbf{(1)}$$

$$= \arg\max_{\pi(c)} \; P(\pi(c)) \prod_{i=1}^{k} \frac{P(M_i(c)|\pi(c))}{P(M_i(c)} \qquad \textbf{(2)}$$

$$= \arg\max_{\pi(c)} \; P(\pi(c)) \prod_{i=1}^{k} P(M_i(c)|\pi(c)) \qquad \textbf{(3)}$$

À partir du jeu de données d'entraînement, nous comptons le nombre de fois que nous avons une correspondance positive pour le constituant c par l'analyseur i. Nous avons aussi besoin du nombre de fois que nous obtenons une correspondance positive pour le constituant c en général.

En prenant compte des statistiques basées sur le jeu de données, nous obtenons l'équation 4 qui suit. Ici, C est l'ensemble des constituants suggérés par les analyseurs. N est la fonction qui retourne le compte de correspondance.

$$P(\pi(c) = t) \prod_{i=1}^{k} P(M_i(c)|\pi(c) = vrai)$$

$$= \frac{N(\pi(c) = vrai)}{|C|} \prod_{i=1}^{k} \frac{N(M_i(c), \pi(c) = vrai)}{N(\pi(c) = vrai)} \qquad \textbf{(4)}$$

La deuxième technique s'appelle la commutation d'analyseurs (*parser switching*). Nous utilisons une sous-méthode appelée commutation de similarité. La similarité est calculée entre un résultat d'analyse avec ceux produits par les autres analyseurs. Ensuite, nous gardons le résultat qui est le plus semblable aux autres. Tout comme pour le vote par constituant, il existe une version probabiliste de cette méthode où nous donnons un poids aux différents analyseurs basé sur une classification faite précédemment.

Henderson et Brill mentionnent aussi qu'il est possible de produire de meilleurs modèles en utilisant des caractéristiques linguistiques des constituants ainsi que leur contexte linguistique. Selon eux, la seule utilité de l'utilisation du contexte serait lorsqu'un constituant

minoritaire trouve la bonne réponse. Un constituant minoritaire est un résultat qui va à l'encontre de la majorité. Il faudrait donc utiliser le contexte pour détecter que ce constituant minoritaire est correct et ainsi renverser le constituant majoritaire. Toutefois, ils disent que pour les caractéristiques qu'ils ont étudiées, aucun gain de performance significatif n'a été détecté.

5.3 Conception de plate-forme de combinaison

Brunet-Manquat ont écrit le premier article que nous avons répertorié qui traite de combinaisons d'analyseurs syntaxiques basés sur les dépendances. En somme, il en vient à utiliser l'hybridation d'analyseurs (pondérée) présentée précédemment.

Par contre, il élabore un peu sur quelques étapes antérieures afin de créer une meilleure correspondance entre les résultats d'analyseurs. Le traitement des sorties des différents analyseurs contient trois grandes étapes : l'apprentissage des indices de confiance, la normalisation et la construction d'analyses de dépendances.

L'étape de l'apprentissage des indices de confiance est indépendante du reste du processus. Effectivement, cette étape consiste à prendre chaque analyseur et de lui attribuer un indice de confiance basé sur ses résultats sur un corpus grâce à l'apprentissage automatisé par classificateur.

Cet indice correspond au calcul de la F-mesure, une moyenne harmonique de la précision et du rappel (deux métriques habituellement utilisées dans les analyseurs syntaxiques basés sur les constituants).

Supposons qu'après l'apprentissage nous avons trois analyseurs, A, B et C, avec les indices suivants (nous utilisons ces analyseurs pour le reste de l'exemple) :

Tableau 5.1 Plate-forme de combinaison : Indices de confiance pour chaque analyseur

Analyseur	Indice
A	0.25
B	0.50
C	0.40

L'étape de normalisation se subdivise en deux phases, soit l'extraction et la projection. Durant la phase d'extraction, nous soutirons des informations syntaxiques à partir de trois types d'analyseurs : les analyseurs syntaxiques basés sur les constituants qui produisent des segmentations en groupes, les analyseurs syntaxiques basés sur les dépendances qui produisent des relations de dépendances et les analyseurs mixtes basés sur les deux structures syntaxiques.

Supposons que les trois analyseurs aient analysé la phrase "Cette$_1$ phrase$_2$ est$_3$ un$_4$ exemple$_{5.6}$" comme illustré dans le Tableau 5.3. Les colonnes contiennent l'identificateur des têtes trouvées par chacun des analyseurs. À titre de rappel, l'identificateur d'un mot correspond à sa position dans la phrase.

Tableau 5.2 Plate-forme de combinaison : Têtes trouvées par chaque analyseur

id	Mot	A	B	C
0	(racine)	-	-	-
1	Cette	1	0	1
2	phrase	3	3	4
3	est	0	0	0
4	un	5	5	5
5	exemple	3	6	6
6	.	3	3	3

Durant la phase de projection, nous prenons ces différentes sorties et nous commençons par les pondérer avec leurs indices de confiance associés à l'analyseur les ayant produites. Ensuite, toutes les sorties sont converties et standardisées en matrices de dépendance.

Désormais, toutes les sorties sont uniformes dans leur représentation et ont un poids différent selon la confiance que nous leur octroyons. Le Tableau 5.3 représente la matrice de dépendance pour l'analyseur A.

Tableau 5.3 Plate-forme de combinaison : Matrice de dépendance

analyseur A	0	1	2	3	4	5	6
0							
1		dét.					
2				sujet			
3	racine						
4						dét.	
5				obj.			
6				ponc.			

L'étape de construction se sépare quant à elle en trois phases, soit la phase de correspondance, la phase de fusion et la phase de production. Dans notre phrase d'exemple, nous n'avons pas besoin de correspondance, mais nous donnons un autre exemple pour démontrer l'utilité de cette étape. Par exemple, un analyseur pourrait traiter le terme « cent vingt-neuf » dans une phrase comme un seul nœud, alors qu'un autre pourrait le diviser en deux nœuds, soit « cent » et « vingt-neuf ». Pour éviter des votes inconsistants, une correspondance est créée entre ces deux nœuds dans un réseau de segmentation unique qui représente la phrase.

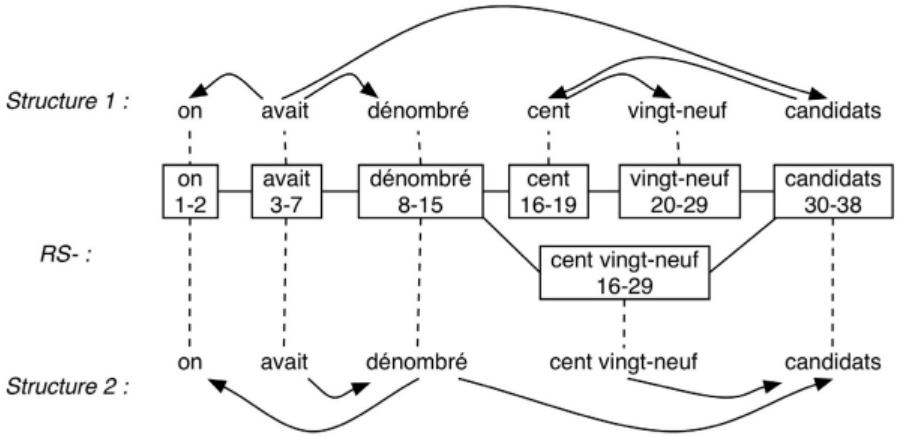

Figure 5.1 Plate-forme de combinaison : Exemple de réseau de segmentation (Brunet-Manquat, 2004)

Durant la phase de fusion, nous calculons des nouveaux indices de confiance pour chaque mot en fonction des indices pour le même mot provenant de chaque analyseur. Ces nouveaux indices de confiance prennent en compte les analyseurs ayant voté pour cette information et ceux qui ont aussi voté contre, affaiblissant ainsi ce nouvel indice.

Ces nouveaux indices correspondent à la somme des poids des analyseurs ayant voté pour diviser par le nombre total d'analyseurs. Le Tableau 5.3 illustre ce calcul pour la phrase d'exemple. Nous construisons ainsi une matrice de fusion pour la nouvelle structure de nœuds avec les indices de confiance recalculés.

Tableau 5.4 Plate-forme de combinaison : Fusion des indices

id	Mot	Tête candidate	Analyseurs	Poids
1	Cette	0	B	(0+0.50+0)/3 = 0.167
		1	A,C	(0.25+0+0.4)/3 = 0.217
2	phrase	3	A,B	(0.25 + 0.50 + 0)/3 = 0.25
		4	C	(0+0+0.4)/3 = 0.133
3	est	0	A,B,C	0.383
4	un	5	A,B,C	0.383
5	exemple	3	A	0.083
		6	B,C	0.300
6	.	3	A,B,C	0.383

La phase de production se base sur les indices de fusion de la phase précédente ainsi que de trois règles fondamentales pour produire une relation de dépendance. Premièrement, une tête est conservée si son indice est supérieur aux indices des autres têtes de dépendance possibles. Deuxièmement, une seule dépendance entre deux mêmes nœuds est conservée, soit celle avec le plus haut indice encore une fois.

Finalement, un nœud ne dépend que d'un seul autre nœud, respectant ainsi une règle fondamentale de la syntaxe par dépendance. Évidemment, seule la dépendance avec le plus haut indice sera gardée. Le résultat final de la phrase d'exemple est présenté dans le Tableau 5.3.

Tableau 5.5 Plate-forme de combinaison : Résultat final

id	Mot	Tête
0	(racine)	-
1	Cette	1
2	phrase	3
3	est	0
4	un	5
5	exemple	6
6	.	3

5.4 Exploration plus approfondie sur les méthodes de combinaison

Zeman et Žabokrtskỳ prennent une approche plus proche de Henderson et Brill, mais conçue pour la combinaison d'analyseurs de dépendance. Plusieurs notions expérimentées ont déjà été expliquées précédemment, comme la commutation de similarité.

Par contre, ils amènent quelques nouveautés, comme explorer la possibilité d'utiliser un plus grand nombre d'analyseurs et de mesurer la contribution d'analyseurs moins performants aux résultats du comité de vote. Par ailleurs, ils essayent à leur tour d'utiliser le contexte dans la prise de décision.

Surtout, ils affirment qu'il faut que chaque analyseur soit assez indépendant des autres pour produire des résultats significatifs dans un comité de vote. Pour ce faire, ils ont pris des mesures durant le développement prouvant que chaque analyseur est le seul à avoir obtenu la réponse correcte dans plusieurs situations. Les résultats de ces mesures les satisfont assez pour affirmer l'indépendance des analyseurs étudiés.

Ils ont fait leur étude sur un ensemble de sept analyseurs de qualités différentes. Après expérimentation, ils en arrivent à la conclusion qu'utiliser les deux meilleurs analyseurs avec deux autres de qualité moyenne forme le meilleur comité de vote possible. La métrique d'évaluation utilisée est l'exactitude, ce qui est environ l'équivalent du compte d'erreurs proposé par Lin.

Pour ce qui est de l'utilisation du contexte (appelé empilage de classificateurs), ils ont eux aussi essayé avec l'approche bayésienne naïve, mais aussi avec les arbres de décision. Ils en sont arrivés à la même conclusion que Henderson et Brill, soit que l'utilisation du contexte, même si elle augmente l'exactitude minimalement, n'a pas beaucoup de potentiel dans l'amélioration des performances d'analyseurs syntaxiques.

5.5 Production d'arbres syntaxiques valides à partir de la combinaison d'analyseurs

Sagae et Lavie reviennent sur les expériences faites par Zeman et Žabokrtskỳ en mentionnant que bien qu'intéressants, les résultats obtenus chutent dramatiquement lorsque les structures sortantes doivent être bien formées. Pour garantir une structure bien formée, les auteurs optent pour une technique qu'ils nomment seconde analyse (*reparsing*).

La première étape de cette technique est de créer un graphe où chaque mot de la phrase représente un nœud du graphe. Ensuite, des arcs doivent être ajoutés pour compléter le graphe. Un arc dirigé représente une relation de dépendance entre deux mots. Chaque sortie d'analyseur contribue en ajoutant son poids aux arcs de dépendance présents dans son propre résultat d'analyse.

Si une seule sortie d'analyseur est bien formée, ceci nous garantit qu'il existe au moins une possibilité de structure bien formée dans l'analyse combinée. Ainsi, une dépendance présente dans plusieurs résultats d'analyse aura un poids plus élevé.

Une fois que le graphe est créé, nous faisons une seconde analyse (reparsing) avec un algorithme comme celui de Chu-Liu/Edmonds ((Yoeng-jin et Tseng-huno, 1965) ; (Edmonds, 1968)) ou bien l'algorithme d'Eisner si nous voulons avoir un résultat projectif (Eisner, 1996) (voir Section 3.2). Ainsi, nous obtenons un nouvel arbre de recouvrement minimal qui théoriquement est bien formé si une des sorties l'était.

Pour décider de la valeur du poids que chaque analyseur peut accorder à un arc, trois approches ont été prises. La première approche est d'octroyer le même poids à tous les analyseurs. La deuxième consiste à donner un poids selon l'exactitude de l'analyseur (calculée selon ses résultats sur la section de développement du Wall Street Journal (WSJ)). Finalement, la troisième approche prend aussi en compte l'étiquette de catégorie syntaxique du mot analysé et accorde un poids en fonction de l'exactitude de l'analyseur pour cette étiquette particulière. La troisième méthode est plus laborieuse, mais a procuré la meilleure amélioration de performance.

5.6 Synthèse de l'état de lart

En résumé, plusieurs chercheurs ont approché de diverses façons la fusion d'analyseurs. Des plateformes expérimentales pour la fusion ont aussi été développées et présentées. Néanmoins, un problème important subvenait lors de la fusion, puisque les arbres syntaxiques produits étaient invalides. C'est pour cette raison que des travaux ont aussi été effectués sur la production d'arbres syntaxiques valides en sortie de la fusion.

L'ensemble de ces expériences a prouvé la faisabilité d'améliorer les performances d'ana-

lyseurs grâce à la fusion, mais nous n'avons pas réussi à recueillir de l'information sur le contexte expérimental. Dans ces articles, le contexte expérimental se révèle très peu défini. Par conséquent, nous accordons beaucoup d'importance à la définition du format de fusion, l'identification des corpus utilisés et sur la sélection d'analyseurs performants.

En effet, dans ces articles, les analyseurs utilisés dans la fusion ne font pas partis des plus performants actuellement. Par conséquent, nous avons donc décidé de faire nos propres expériences avec les meilleurs outils disponibles et dans un environnement expérimental bien défini, tel que mentionné dans la Section 2.3.

CHAPITRE 6

APPROCHE PROPOSÉE POUR LA FUSION ET LA RECONSTRUCTION D'ARBRES

Dans cette section, nous présentons les grandes lignes de l'approche prise en fonction de l'état de l'art. Toutefois, pour les précisions plus techniques de l'expérimentation, les détails seront donnés dans le chapitre traitant de la méthodologie.

En premier lieu, pour mesurer les performances générales des analyseurs, nous nous inspirons des métriques simples inspirées par Lin. En effet, ce que Lin appelait *error count*, nous l'appelons dans nos travaux UAS, terminologie empruntée aux outils d'évaluation de la campagne CoNLL. En deuxième lieu, basé sur l'article de Henderson et Brill, nous utilisons des techniques de combinaison basées sur un système de votes, quoique ces derniers l'ont fait sur une syntaxe représentée par constituants.

Toutefois, en s'inspirant de la reprise de ces travaux par Zeman et Žabokrtskỳ, nous prenons cette approche de vote, mais dans un contexte de représentation par dépendance. En plus, Zeman et Žabokrtskỳ suggèrent de varier les types d'analyseurs présents dans le comité de vote afin d'avoir des résultats plus indépendants, améliorant ainsi l'effet combinatoire.

Par conséquent, nous utilisons une nouvelle approche de variété dans le comité, soit en prenant des analyseurs se basant sur des algorithmes d'analyse différents. Nous voulons assurer cette variété en utilisant des analyseurs basés sur les transitions, des analyseurs basés sur les graphes ainsi que des analyseurs basés sur la grammaire.

Tout comme dans les travaux de Zeman et Žabokrtskỳ, nous calculons aussi les taux de confiance selon le contexte linguistique en considérant la catégorie syntaxique du mot analysé.

Nous utilisons aussi les étapes de conception de plate-forme suggérées dans les travaux de Brunet-Manquat. La première étape consiste en l'apprentissage des indices de confiance. Pour ce faire, nous faisons analyser un corpus non-annoté par chaque analyseur, nous l'évaluons par rapport au corpus annoté et les résultats de UAS sont conservés ainsi que les pointages par catégorie syntaxique pour chacun.

L'étape suivante consiste en l'étape de normalisation. La première phase de cette étape est l'extraction. Nous prenons donc les résultats sous le format de chaque analyseur afin d'en extraire de l'information syntaxique selon la représentation de notre choix. Nous tombons par la suite dans la phase de projection. À l'information syntaxique extraite de ces analyseurs, nous ajoutons les indices de confiance associés aux résultats.

Nous amorçons ensuite l'étape de construction par la phase de fusion. Dans la phase de fusion, les indices de confiance sont recalculés en fonction des différents analyseurs ayant trouvé la même tête pour un mot. Ensuite, dans la phase de production, nous identifions quelle tête a le plus haut taux de confiance et nous l'élisons comme tête gagnante dans l'analyse combinée.

6.1 Système de fusion

L'idée générale du système de fusion pour une phrase est d'utiliser plusieurs analyseurs performants pour produire une seule analyse unifiée plus précise. Pour ce faire, nous faisons analyser la phrase par chaque analyseur et nous conservons le résultat localement. Pour chaque mot de la phrase originale, nous extrayons ensuite une *production*. Selon la technique de fusion et l'analyseur, un poids différent sera accordé pour chaque production. Le système de votes fusionnera les productions similaires et leur accordera un poids plus lourd selon la technique de votes utilisée. La production fusionnée avec le plus gros poids sera la production conservée.

6.1.1 Algorithme général

Soit W un mot d'une phrase à analyser. Ce mot est représenté par sa forme et sa catégorie syntaxique. Sa tête et sa relation de dépendance seront ajoutées à sa représentation une fois analysé.

Soit *Prod* une production générée par un analyseur pour un mot. Cette sortie contient le quadruplet <*mot, tête, relation de dépendance, analyseur*>, soient le mot W, la tête du mot analysé, la relation de dépendance l'associant à cette tête et l'analyseur courant ayant produit cette sortie.

Soit A un analyseur.

Soit $ParsersResults$ un tableau de deux dimensions qui contient le résultat d'analyse pour chaque mot par chaque analyseur.

Soit `ANALYZE`(S_0, A) la fonction qui analyse la phrase S_0 par l'analyseur A. Pour chaque mot W de la phrase S_0, `ANALYZE` produit une production $Prod$.

Soit `VOTE`($Prod$) la fonction qui place un vote pour la tête du mot en conservant l'analyseur l'ayant placé ainsi que la relation de dépendance. Cette fonction prend en entrée une production $Prod$ et génère comme sortie un vote V, qui est constitué de la production $Prod$ accompagnée de son poids de vote.

Soit `COUNT`($Ballot$) la fonction qui compte les votes et retourne la tête de dépendance avec le plus haut taux de confiance. Les différentes variantes de la technique de fusion reposent précisément sur la différence d'implémentation de cette fonction.

Soit `GetDeprel`($Head$, $Ballot$) la fonction qui prend en entrée la tête de dépendance gagnante et le ballot de votes. Cette fonction retourne la relation de dépendance entre le mot et sa tête. Nous cherchons à récupérer la relation de dépendance trouvée par le meilleur analyseur ayant voté pour cette tête.

$S_0 \leftarrow [W_1, W_2, ..., W_n]$
$Committee \leftarrow \{A_1, A_2, ..., A_m\}$

`FUSION`(S_0, Committee) :
$S_F \leftarrow \emptyset$
$ParserResults \leftarrow \emptyset$

pour $i \leftarrow 0$ a taille[Committee]
    $ParserResults[i] \leftarrow$ `ANALYZE`(S_0, A_i)
pour $i \leftarrow 0$ a taille[S_0] :
    $W_0 \leftarrow S_0[i]$
    $Ballot \leftarrow \emptyset$
    pour $j \leftarrow 0$ a taille[Committee] :
        $Prod_{A_j} \leftarrow ParserResults[j][i]$
        $V \leftarrow$ `VOTE`$(Prod_{Aj})$
        $Ballot \leftarrow Ballot \cup V$

$Head \leftarrow$ `COUNT`$(Ballot)$
$Deprel \leftarrow$ `GetDeprel`$(Head, Ballot)$
$W_F \leftarrow W_0 \cup \{Head, Deprel\}$
$S_F \leftarrow S_F \cup \{W_F\}$
return S_F

6.1.2 Taux de confiance

Trois variantes existent pour le calcul du taux de confiance pour chaque vote, soit la technique de vote à poids égal, de vote pondéré par LAS/UAS et vote pondéré par POS. Les deux dernières variantes sont basées sur des lois probabilistes.

6.1.3 Comité de vote

Le comité de vote est en fait l'ensemble des analyseurs ayant à prendre une décision commune. Nous empruntons la terminologie d'un vrai système de votes afin de faciliter l'analogie. Chaque membre du comité s'exprime à tour de rôle de façon indépendante pour la tête candidate de son choix.

6.1.4 Techniques de vote

Comme il a été brièvement mentionné précédemment, la principale différence entre ces techniques consiste principalement en la façon de pondérer les votes produits de chaque analyseur.

La technique de vote à poids égal consiste à attribuer un droit de vote de poids égal à chaque analyseur. Chaque analyseur place son vote et ensuite, le candidat ayant le plus de votes gagne.

Dans la stratégie de **vote pondéré par efficacité de l'analyseur** (vote pondéré par UAS), au lieu d'attribuer un poids égal de vote pour chaque membre du comité, le vote de chaque analyseur est pondéré en fonction de sa performance, calculée préalablement à partir de statistiques obtenues en l'appliquant à un corpus.

Nous supposons que la probabilité que l'analyseur génère la bonne réponse est proportionnelle à son efficacité. Deux variantes de cette technique existent, soit la pondération probabiliste basée sur la métrique du UAS et la pondération basée sur le LAS.

Le taux de confiance est ensuite calculé en appliquant la loi fondamentale utilisée pour multiplier les probabilités soit :

$TC_{candidat} = 1 - \prod_{i=0}^{n} (1\text{-}P_i)$, où n est le nombre d'analyseurs supportant ce candidat et P_i le UAS de l'analyseur i.

L'approche de **vote pondéré par efficacité de l'analyseur en fonction de la catégorie syntaxique** (vote pondéré par POS) est encore un raffinement d'une technique présentée. Cette fois, au lieu de pondérer selon le UAS de l'analyseur, nous pondérons en fonction du libellé de catégorie syntaxique du mot analysé. Effectivement, grâce aux méthodes d'évaluation, nous avons la précision de l'analyseur pour chaque type de libellé de catégorie syntaxique et nous pouvons donc avoir une pondération spécialisée en fonction du contexte. Le taux de confiance est calculé selon la même loi probabiliste présentée précédemment, soit :

$TC_{candidat} = 1 - \prod_{i=0}^{n} (1\text{-}P_i)$, où n est le nombre d'analyseurs supportant ce candidat. et P_i la précision de l'analyseur i pour la catégorie syntaxique du mot.

6.2 Reconstruction d'arbre

La méthodologie de vote mot par mot semble amener des résultats intéressants. Par contre, cette approche ne garantit pas un arbre de dépendance valide à la sortie. Effectivement, en ne traitant pas l'ensemble de la phrase à la fois et en isolant chaque mot de son contexte, nous pouvons prendre des décisions qui violent les propriétés fondamentales de l'arbre en cours de construction. Pour éviter de produire un arbre incorrect, nous vérifions la validité de ce dernier après sa production et, au besoin, nous le reconstruisons. Les techniques de reconstruction sont présentées à la Section 6.2.2.

6.2.1 Vérification d'arbres

Pour vérifier la validité de l'arbre, nous observons le respect des trois propriétés suivantes :

1) L'arbre doit avoir une seule racine.
Comme il a été mentionné précédemment, une racine artificielle est créée pour s'assurer que chaque mot soit connecté à un autre. Toutefois, si cette racine artificielle a plus d'un fils, cela indique que l'analyse retournée n'est pas représentée par un arbre unique qui couvre toute la phrase.

Afin de valider le respect de cette règle, tous les mots de la phrase sont parcourus séquentiellement, en mémorisant les cas où la tête du mot est la racine artificielle. Il suffit alors de vérifier que celle-ci n'est apparue qu'une seule fois.

2) Chaque noeud doit avoir un et un seul parent.
Il est facile de vérifier que, selon notre algorithme de vote, cette situation ne peut pas se produire.

3) L'arbre doit être acyclique.
Puisque le parent de chaque nœud a été voté indépendamment, nous ne pouvons prédire si ceci créera un cycle dans l'analyse résultante. Par exemple, prenons trois mots d'une phrase aux positions 6,7 et 8. On pourrait voter indépendamment que le mot 7 est le parent du mot 6, le mot 8 le parent du mot 7 et le mot 6 est le parent du mot 8. Nous aurions donc la présence d'un cycle.

Pour vérifier la présence d'un cycle, nous identifions la liste des ancêtres pour chaque mot. Nous commençons à peupler cette liste en prenant le premier parent du nœud. Si le même nœud se retrouve deux fois dans la liste d'ancêtres, en d'autres mots, si un ancêtre est un ancêtre à lui-même, alors il y a présence d'un cycle.

6.2.2 Stratégies de reconstruction

Si l'arbre se trouve à être invalide, nous nous devons de traiter la phrase à nouveau afin de produire un arbre correct. Nous avons décidé d'explorer deux approches. En premier lieu, nous prenons tout simplement l'analyse produite par le meilleur analyseur, qui lui garantit un arbre correct. En deuxième lieu, nous prenons l'approche proposée par Sagae et Lavie,

soit la reconstruction d'arbre par seconde analyse.

Reconstruction par analyse du meilleur analyseur

L'avantage de cette approche est qu'elle garantit une production d'arbre valide. Ainsi, même si nous perdons l'amélioration potentielle de la combinaison d'analyseurs pour ces analyses, utiliser cette approche pour les quelques arbres incorrects est envisageable. En effet, comme nous le verrons plus loin, la proportion de cas où l'analyse résultante n'est pas un arbre valide est plutôt faible.

Reconstruction par MST

La première étape de cette technique est de créer un graphe où chaque mot de la phrase est un nœud. Prenons pour exemple la phrase "The finger-pointing has already begun.". Nos noeuds seront identifiés comme suit :

Tableau 6.1 Reconstruction par MST : Noeuds du graphe pour la phrase "The finger-pointing has already begun."

id	Mot
0	ROOT
1	The
2	finger
3	-
4	pointing
5	has
6	already
7	begun
8	.

Par la suite, nous construisons un tableau contenant, pour chaque mot, la tête proposée par chaque analyseur, ce qui donne un résultat comme celui illustré au Tableau 6.2.

Tableau 6.2 Reconstruction par MST : Résultats d'analyse individuels de chaque analyseur pour la phrase "The finger-pointing has already begun".

id	Mot	DeSR	Ensemble	IDP	LTH	MST	Stanford
0	ROOT	-	-	-	-	-	-
1	The	0	2	4	4	4	4
2	finger	4	5	4	4	4	4
3	-	2	5	2	2	2	4
4	pointing	5	5	5	5	5	5
5	has	1	0	0	0	0	0
6	already	5	5	5	5	5	5
7	begun	5	5	5	5	5	5
8	.	5	5	5	5	5	5

Ensuite, chaque sortie d'analyseur contribue en ajoutant son poids aux arcs de dépendance présents dans son propre résultat d'analyse. Pour combiner le poids des analyseurs supportant le même arc, nous utilisons la formule suivante :

P_{arc} = 1 - $\prod_{i=0}^{n}$ (1-P$_i$), où n est le nombre d'analyseurs supportant ce candidat et P_i le poids de l'analyseur supportant le mot.

Le poids de l'analyseur correspond à son UAS. Le résultat des arcs différents par noeuds est donné dans la table suivante, suivant les UAS qui seront donnés ultérieurement pour chaque analyseur.

Tableau 6.3 Reconstruction par MST : Arcs du graphe pour la phrase ”The finger-pointing has already begun.”

id	Mot	Arc	Poids
1	The	(1,0)	0,89
		(1,2)	0,86
		(1,4)	0,99
2	finger	(2,4)	0,99
		(2,5)	0,86
3	-	(3,2)	0,99
		(3,4)	0,75
		(3,5)	0,86
4	pointing	(4,5)	1,00
5	has	(5,0)	0,89
		(5,1)	0,99
6	already	(6,5)	1,00
7	begun	(7,5)	1,00
8	.	(8,5)	1,00

Si une seule sortie d’analyseur est bien formée, ceci nous garantit qu’il existe au moins une possibilité de structure bien formée dans l’analyse combinée. Ainsi, plus une dépendance est présente dans plusieurs résultats d’analyse, plus son poids sera élevé. Ces noeuds et arcs produisent alors le graphe représenté dans la Figure 6.1 :

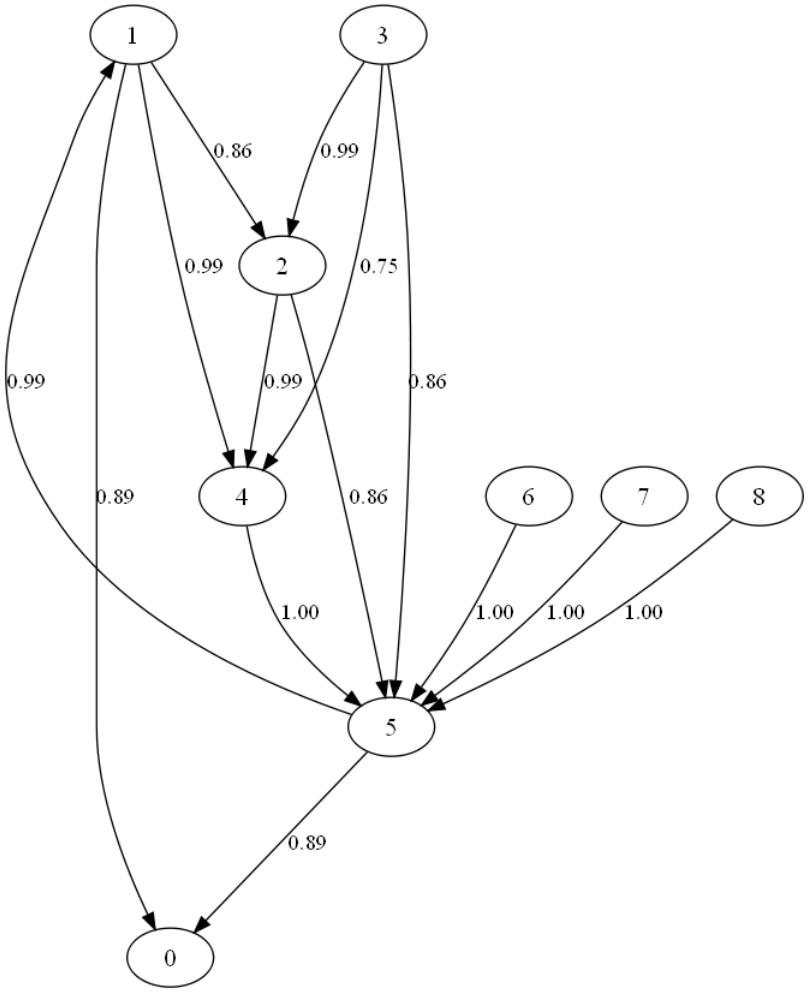

Figure 6.1 Reconstruction par MST : Graphe avec le poids des arcs combinés

À partir du graphe obtenu, nous faisons une seconde analyse (*reparsing*) avec un algorithme d'analyse de dépendance classique pour trouver le MST. Dans notre cas, nous utilisons l'algorithme d'Edmonds, dont les grandes lignes ont été présentées précédemment.

6.3 Sommaire général de l'analyse par fusion

La phrase en entrée est tout d'abord analysée par chaque analyseur du comité. Chaque mot de la phrase est pondéré en fonction de la technique de fusion choisie et de la performance de l'analyseur. Une fois les analyses pondérées, les analyses communes pour un même mot sont fusionnées et leurs poids sont combinés. Ensuite, une seule tête est élue pour chaque mot de la phrase.

Finalement, une validation est faite pour s'assurer que la phrase produite est un arbre syntaxique correct et, au besoin, l'arbre est reconstruit selon la technique de reconstruction choisie.

CHAPITRE 7

MÉTHODOLOGIE

7.1 Choix des analyseurs

Dans cette section, nous présentons les travaux qui ont été effectués pour répondre à l'objectif O1, soit de trouver les meilleurs analyseurs existants.

7.1.1 Catégorisation des analyseurs trouvés et élimination

Plusieurs analyseurs ont été étudiés au courant de nos expériences. Nous pouvons les catégoriser en quatre groupes ; les analyseurs qui n'étaient pas basés sur les dépendances et qui auraient nécessité la création de passerelles ou transformations pour les utiliser. Ensuite, il y a quelques analyseurs que nous n'avons pas réussi à déployer ou dont le format était non standard et très difficilement adaptable. Nous avons évité de travailler avec ces deux groupes d'analyseurs, car nous aurions pu induire des sources d'erreurs supplémentaires dans la passerelle d'équivalence et notre objectif est justement de diminuer le taux d'erreur de l'analyse syntaxique.

Parmi les analyseurs qui respectent des formats standards, il y a ceux qui fonctionnent avec le format CoNLL et ceux qui adoptent le format Stanford. Nous avons adopté le format CoNLL et avons tenté une passerelle de Stanford vers CoNLL. Parmi les analyseurs respectant ces deux standards, certains n'étaient tout simplement pas assez performants et sont tombés dans le troisième groupe d'analyseurs, soit les analyseurs avec des résultats trop faibles pour faire partie du comité de vote. Finalement, le quatrième groupe est constitué des analyseurs restants avec des performances satisfaisantes pour faire partie du comité, soit 75% et plus de UAS.

7.1.2 Analyseurs choisis

Comme présenté dans le Chapitre 3, les analyseurs syntaxiques par dépendance peuvent être classifiés en trois grandes familles, soit les analyseurs basés sur les transitions, ceux basés sur les graphes et ceux basés sur les grammaires. Si nous tenons compte des analyseurs que

par leurs performances, les meilleurs analyseurs sont majoritairement basés sur les transitions. Nous avons donc ensuite cherché les meilleurs analyseurs basés sur les grammaires et sur les graphes afin d'apporter de la variété algorithmique à notre comité. Nous nous retrouvons donc avec de multiples analyseurs basés sur les transitions, puisqu'ils sont très performants, et nous cherchons à garder ceux dont la contribution au comité de vote est significative.

MaltParser (Nivre *et al.*, 2007b) est un analyseur développé par Johan Hall, Jens Nilsson et Joakim Nivre. Cet analyseur offre la possibilité d'analyser une phrase grâce à neuf variantes algorithmiques de la technique shift-reduce. Il utilise l'apprentissage machine pour entrainer l'analyseur sur une banque d'arbres (corpus) et pour décider quelle action prendre à une étape non-déterministe, comme expliqué plus tôt dans la section présentant les analyseurs basés sur les transitions.

MaltParser n'a pas été sélectionné pour notre comité de vote. L'analyseur Ensemble (Surdeanu et Manning, 2010), utilisé dans nos expérimentations, est basé sur le MaltParser. Effectivement, Ensemble, développé par Mihai Surdeanu, interpole linéairement une analyse de phrase à partir de l'analyse produite par les différentes variantes algorithmiques du MaltParser. En d'autres mots, il lance plusieurs instances du MaltParser, mais avec les différentes variantes algorithmiques, prend les multiples sorties et interpole une analyse.

L'analyseur Stanford (Klein et Manning, 2003), crée par Klein et Manning, est notre seul analyseur basé sur les grammaires. Il utilise les grammaires non-contextuelles stochastiques pour résoudre l'analyse syntaxique. Il a été lui aussi entraîné sur le Penn Treebank.

MSTParser (McDonald *et al.*, 2006) est le seul analyseur présent dans nos expérimentations qui est basé sur les graphes. Il cherche l'arbre sous-tendant maximal sur des graphes dirigés, un peu comme présenté dans notre cadre théorique. Cet analyseur a été produit par Ryan McDonald, de Google Research.

DeSR (Attardi, 2006), développé par Giuseppe Attardi à l'University of Pisa, est un autre analyseur basé sur les transitions. Celui-ci offre la possibilité d'entraîner l'analyseur avec différents classificateurs afin d'inférer différents modèles pour une même banque d'arbres. Nous avons utilisé le modèle fourni sur leur site puisque nous supposons que c'est le modèle suggéré pour obtenir les résultats expérimentaux présentés sur le site, ce qui était le cas.

Le ISBN Dependency Parser (Titov et Henderson, 2007), ou encore IDP, est produit

conjointement par Ivan Titov de University of Genova et James Henderson de University of Edinburgh. Cet analyseur est lui aussi basé sur les transitions. Celui-ci se différencie par sa technique d'entraînement, puisqu'il classifie les banques d'arbres en se basant sur les Incremental Sigmoid Belief Networks (ISBN), d'où le nom.

LTH (Johansson et Nugues, 2007) a été développé à Lund University par Richard Johansson. Cet outil permet l'analyse sémantique d'un texte. Il est aussi un analyseur syntaxique basé sur les transitions puisqu'il utilise en partie MaltParser. L'entraînement statistique a été effectué sur Penn Treebank. Ses résultats sont très intéressants, comme nous le verrons plus tard.

7.2 Choix du format

Tel que mentionné dans l'objectif O1, un format doit être fixé afin d'uniformiser les sorties d'analyseurs. Dans la présente section, nous exposons le format que nous avons choisi.

7.2.1 Présentation du format

Le format CoNLL (Nivre, J., 2007) représente une phrase sous la forme d'un tableau avec plusieurs champs détaillant le contexte syntaxique d'un mot. Les champs les plus importants dans le cadre de notre travail sont ID, spécifiant la position du mot dans la phrase, FORM, représentant le mot en tant que tel, POSTAG, représentant le libellé de catégorie syntaxique du mot, HEAD, représentant le ID de la tête dont le mot est dépendant et finalement DEPREL, représentant le type de relation de dépendance entre le mot et sa tête.

Nous montrons ici en exemple la représentation sous le format CoNLL de la phrase "The economy's temperature will be taken from several vantage points this week, with readings on trade output, housing and inflation."

Tableau 7.1 Exemple de représentation de phrase sous le format CoNLL

ID	FORM	POSTAG	HEAD	DEPREL
1	The	DT	2	NMOD
2	economy	NN	4	NMOD
3	's	POS	2	SUFFIX
4	temperature	NN	5	SBJ
5	will	MD	0	ROOT
6	be	VB	5	VC
7	taken	VBN	6	VC
8	from	IN	7	ADV
9	several	JJ	11	NMOD
10	vantage	NN	11	NMOD
11	points	NNS	8	PMOD
12	this	DT	13	NMOD
13	week	NN	7	TMP
14	,	,	7	P
15	with	IN	7	ADV
16	readings	NNS	15	PMOD
17	on	IN	16	NMOD
18	trade	NN	17	PMOD
19	,	,	18	P
20	output	NN	18	COORD
21	,	,	20	P
22	housing	NN	20	COORD
23	and	CC	22	COORD
24	inflation	NN	23	CONJ
25	.	.	5	P

Le format Stanford représente environ la même information, mais présentée différemment. Nous allons utiliser le même formalisme que pour CoNLL pour expliquer la structure. Au lieu d'orienter la structure autour des mots, la syntaxe du format est plutôt basée sur les DEPREL. Effectivement, chaque ligne des résultats générés est du format suivant : DEPREL(HEAD-ID, FORM-ID)

En gardant la même phrase comme exemple, voici sa représentation sous le format Stanford :

det(economy-2, The-1)
poss(temperature-4, economy-2)

nsubjpass(taken-7, temperature-4)
aux(taken-7, will-5)
auxpass(taken-7, be-6)
amod(points-11, several-9)
nn(points-11, vantage-10
prep_from(taken-7, points-11)
det(week-13, this-12)
tmod(taken-7, week-13)
prep_with(taken-7, readings-16)
prep_on(readings-16, trade-18)
prep_on(readings-16, output-20)
conj_and(trade-18, output-20)
prep_on(readings-16, housing-22)
conj_and(trade-18, housing-22)
prep_on(readings-16, inflation-24)
conj_and(trade-18, inflation-24)

Comme pour toute analyse syntaxique, tout mot n'a qu'une seule tête. Nous nous attendons à ce qu'il y ait le même nombre de relations de dépendance produites que le nombre de mots. Finalement, notre choix s'est arrêté sur le format CoNLL puisque la plupart des analyseurs sélectionnés sont compatibles avec ce format.

7.2.2 Uniformisation des analyseurs

Si la phrase en entrée est une phrase écrite, elle est préalablement transformée en format CoNLL grâce à un outil fourni par l'analyseur LTH. Chaque analyseur a un format d'entrée lui étant spécifique. Souvent, il est facile de transformer le format CoNLL en ce format particulier. Il suffit d'ajouter, supprimer ou changer l'ordre des colonnes en entrée. Par contre, l'analyseur MSTParser prend un tout autre format, mais il peut tout de même être converti au format CoNLL.

En sortie, chaque analyseur produit aussi une sortie qui lui est spécifique, mais encore une fois, il n'est pas difficile de générer un fichier sous le format CoNLL à partir de leurs formats spécifiques. Seul Stanford demande un peu plus de travail. Ce dernier produit des relations de dépendance qui ne sont pas similaires à celles produites par les autres analyseurs, alors une table de correspondance est nécessaire pour faire le passage. Pour des fins de comparaison,

nous avons converti toutes les analyses produites sous le format suivant :

ID FORM FORM _ POSTAG FORM FORM POSTAG HEAD DEPREL _

Ce format correspond aussi au corpus étalon, qui nous servira à évaluer la performance de l'analyseur.

7.2.3 Choix du corpus

Comme mentionné dans l'objectif O4, nous avons besoin d'un jeu de données pour effectuer nos expériences. Ce jeu de données provient du corpus que nous avons sélectionné, qui est présenté dans la présente section.

7.2.4 Campagnes d'évaluation CoNLL

Le NLL (pour *natural language learning*) est une branche de la linguistique informatique. Au cours des dernières décennies, plusieurs groupes de travail, ateliers et conférences ont traité du NLL, mais il n'y avait pas de constance dans la nomenclature ou d'organisme encadrant ces activités. De plus, les thématiques des activités étaient décentralisées et portaient sur toutes sortes de sujets, dont l'apprentissage-machine, le connexionnisme et les ontologies.

Finalement, les principaux acteurs du domaine ont décidé de créer une filière de l'ACL (Assocation of Computational Linguistics) s'appelant le SIGNLL (Tjong, E., 2011) (Special Interest Group on Natural Language Learning). Ce groupe focalise ses activités autour de l'apprentissage-machine dans le domaine du NLL, mais il explore et applique aussi des techniques d'autres branches pour résoudre leurs problèmes.

La campagne CoNLL, soit la Conference on Computational Natural Language Learning, est la rencontre annuelle du SIGNLL, regroupant des spécialistes internationaux dans le domaine du traitement informatique de l'apprentissage de la langue naturelle. Chaque année, durant la campagne, une tâche commune est définie sur laquelle tous les chercheurs intéressés peuvent travailler. Cette tâche représente toujours un sujet d'actualité dans le domaine sur lequel la communauté veut faire des avancements.

Les tâches communes de la campagne 2006 et 2007 se sont concentrées sur différentes facettes de l'analyse syntaxique par dépendance, comme l'adaptation aux domaines spécifiques

et à l'aspect multilingue en 2007 (Nivre *et al.*, 2007a). Les résultats de ces campagnes ont été une forte source d'inspiration dans la recherche d'analyseurs syntaxiques par dépendance performants puisque le classement général des résultats des analyseurs par auteur est disponible. Effectivement, une bonne partie de l'état de l'art consulté et des analyseurs utilisés proviennent de travaux faits pour la campagne CoNLL.

7.2.5 Corpus CoNLL

Nous avons utilisé le corpus de la campagne CoNLL 2008, soit le corpus WSJ. Ce corpus est principalement constitué du Penn Treebank, complémenté du contenu sémantique de NomBank et PropBank (Surdeanu *et al.*, 2008).

La section d'entraînement, la section *train*, contient les sections 01-22 du Penn Treebank. La section de développement, la section *dev*, contient la section 24 du Penn Treebank. La section *test* contient la section 23 du Penn Treebank.

Étant donné que la section d'entraînement a été utilisée dans l'entraînement de quelques analyseurs testés, nous ne pouvons l'utiliser pour entraîner notre système, puisque le système serait biaisé. Nous ne pouvons donc pas utiliser la structure usuelle du corpus dû à ce biais. Nous avons donc pris la section de développement et la section de tests ensemble et l'avons subdivisé en trois corpus, le corpus A, contenant 1200 phrases, le corpus B, contenant 1199 phrases et le corpus C, contenant 1344 phrases.

Puisque nous n'avons pas un aussi grand échantillon de phrases que la section d'entraînement, nous allons utiliser la validation croisée pour éliminer le biais expérimental possible durant nos observations tout en gardant la possibilité de les valider sur un autre corpus.

7.3 Méthodes d'évaluation sélectionnées

L'objectif O3 consiste à choisir une métrique afin d'évaluer la performance des analyseurs. Notre choix s'est arrêté sur les méthodes d'évaluation instaurées par la campagne CoNLL. Nous générons donc plusieurs statistiques intéressantes comme le UAS, le LAS en plus de plusieurs statistiques détaillées comme la précision par catégorie syntaxique (POS). Nous utilisons le UAS basé sur les mots, comme mentionné précédemment, ainsi que la précision par POS dans nos techniques de fusion.

7.4 ComboParser

ComboParser est le nom de notre analyseur basé sur la fusion. Cette section est consacrée à expliquer son fonctionnement en détail ainsi qu'à comprendre les informations générées par cet analyseur.

7.4.1 Entrées et sorties de comboParser

Le comboParser prend en entrée les analyses produites par les différents analyseurs. Ces analyses doivent être uniformisées sous le format spécifié ci-dessus. La sortie respecte aussi le format spécifié ci-dessus. ComboParser prend le corpus identifié dans la configuration en entrée, annoté par les différents analyseurs. L'analyseur prend aussi en entrée un oracle pour comparer ses résultats et produire des statistiques sur l'analyse en cours. En sortie, comboParser produit les phrases annotées, des statistiques sur ses performances et un journal de bord pour décrire le contexte quand il n'arrive pas à la bonne réponse.

7.4.2 Exécution de comboParser

Dans cette sous-section, nous discutons du plan d'expérience. Ceci consiste en la configuration de l'expérience à faire, l'exécution des expériences suivie de la production des statistiques et du journal de bord.

Configuration du contexte expérimental

La configuration du contexte expérimental consiste en l'ensemble des propriétés régissant l'expérimentation. Par exemple, nous devons entre autres fixer la technique de vote, la technique de reconstruction et les analyseurs faisant partie du comité. La table suivante décrit toutes les valeurs possibles pour les paramètres d'expérimentation.

Tableau 7.2 Paramètres de configuration de comboParser

Paramètre	Valeurs
Nom du corpus	A B C
Taille du corpus	(Variable)
Mode	test production
Stratégie	Vote Vote pondéré par UAS Vote pondéré par POS
Reconstruction	Aucune Meilleur analyseur Arbre de recouvrement minimal
Analyseurs	3 à 6 analyseurs parmi : Stanford Ensemble IDP DeSR MST LTH

Les stratégies de vote et de reconstruction ainsi que les analyseurs ont déjà été présentés en détail précédemment. Le nom du corpus sert à informer l'analyseur du corpus en cours d'utilisation afin de savoir quel corpus étalon il doit utiliser comme référence. Le mode consiste à indiquer à l'analyseur s'il est en mode expérimental ou en mode de production.

Il est à noter que les performances des analyseurs sont conservées dans un autre fichier de description contenant le LAS, le UAS et la précision par catégorie syntaxique (POS) pour chaque analyseur.

Statistiques et journal de bord

Dans les statistiques produites à la fin de chaque expérimentation, les résultats sont classés en fonction des libellés de catégorie syntaxique. Nous identifions d'abord le nombre de fois où l'analyseur s'est trompé pour les mots de ce type de catégorie syntaxique. Le nombre de fois où chaque analyseur a placé un vote minoritaire correct lorsque comboParser s'est trompé est aussi calculé. Une autre statistique calculée est le nombre de fois où n analyseurs

avaient un vote minoritaire correct pour la catégorie syntaxique en question. Voici un extrait des statistiques pour faciliter la compréhension :

JJ : Error rate : 166/3041 (5,46%)

Correct minority votes per parser for POS(JJ) :

---

stanford(82.0) : 36/166 (21,69%)
lth(94.0) : 12/166 (7,23%)
idp(92.0) : 17/166 (10,24%)
desr(94.0) : 23/166 (13,86%)

Nb. of times that 1 out of 4 voters had a correct minority vote : 64/166 (38,55%)
Nb. of times that 2 out of 4 voters had a correct minority vote : 24/166 (14,46%)

Dans cet exemple, JJ est le nom de la catégorie syntaxique. Le taux d'erreur est de 5,46% pour cette catégorie. Pour les votes minoritaires par analyseur, prenons Stanford comme exemple. La valeur de 82,0 représente une précision de 82% pour cet analyseur pour le libellé JJ.

36/166 représente la proportion de cas où que Stanford a identifié la bonne tête pour un mot de la catégorie JJ. En d'autres mots, pour les 166 fois que ComboParser a produit une erreur pour la catégorie syntaxique JJ, Stanford avait la bonne réponse 36 fois.

Ensuite, nous retrouvons le nombre de fois que n analyseurs avaient un vote minoritaire correct. Étant donné que nous avons 4 analyseurs, le nombre maximal de votes minoritaires est de 2, sinon nous aurions majorité. Donc, prenons la 2e ligne de cette catégorie de statistique, soit "Nb. of times that 2 out of 4 voters had a correct minority vote : 24/166 (14,46%) ". En d'autres mots, ceci signifie que pour les 166 fois que ComboParser s'est trompé pour la catégorie syntaxique JJ, il y a 24 cas où 2 analyseurs avaient la bonne réponse.

De plus, des statistiques générales sont compilées par rapport à la reconstruction de phrases. Nous cherchons plus spécifiquement à savoir combien de fois les analyses ne sont pas des arbres valides, la source de l'erreur (plusieurs racines, présence de cycle ou plusieurs

parents) et le taux de succès de reconstruction. Ces statistiques sont produites sous le format suivant :

RECONSTRUCT STATS

BADTREES :111
MULTIROOTS :40
MULTIPARENTS :0
CYCLICTREES :71
RECONSTPCT :100.0%

BADTREES est la statistique indiquant le nombre de fois où l'arbre produit par Combo-Parser est un arbre invalide. Dans l'exemple ci-dessus, 111 arbres sur 2400 étaient invalides. Les trois lignes suivantes indiquent la raison pour laquelle ces 111 arbres étaient invalides. 40 fois sur 111, la raison était qu'il y avait plus d'une racine dans la phrase (MULTIROOTS).

Les 71 autres fois étaient dues au fait qu'il y a un cycle, rendant l'arbre invalide (CYCLICTREES). Aucun arbre n'était erroné pour cause de parents multiples pour un noeud (MULTIPARENTS). Finalement, nous indiquons le pourcentage de fois que nous avons réussi à reconstruire l'arbre (100% dans l'exemple présent).

Un autre ensemble de statistiques qui nous est très utile pour notre analyse des résultats est le nombre de fois où n votes minoritaires corrects ont été placés, où n appartient à [1, taille du comité - 1]. Voici un exemple pour un comité de 4 analyseurs :

MINORITY VOTES STATS

Total error count : 4389
1 correct minority votes occurence : 1996
2 correct minority votes occurence : 1022
3 correct minority votes occurence : 0

Le dernier bloc de statistiques qui nous intéresse pour ce fichier est le nombre de vote minoritaires corrects émis par analyseur, afin de pouvoir tirer des conclusions sur l'influence des analyseurs sur le comité.

PARSER STATS

Total error count : 4389
Correct minority votes for parser stanford : 1055
Correct minority votes for parser lth : 587
Correct minority votes for parser idp : 621
Correct minority votes for parser desr : 755

Ces deux blocs de statistiques ne font que compiler les statistiques de votes minoritaires déjà présentées dans les blocs précédents.

Dans le journal de bord, nous retrouvons une entrée seulement pour les mots dont l'analyse est erronée. Premièrement, nous identifions à quelle phrase appartient ce mot dans le corpus et son index dans celle-ci. Deuxièmement, nous identifions la tête et la catégorie syntaxique du mot erroné. Troisièmement, nous indiquons le nombre de votes minoritaires corrects qui ont été ignorés pour le mot erroné. Nous complétons cette information avec la liste des analyseurs ayant placé ces votes minoritaires. Finalement, nous donnons de façon détaillée la tête identifiée par chaque analyseur pour ce mot. Encore une fois, un extrait d'un journal de bord est de mise par fin de clarification :

ERROR FOR SENTENCE(3) WORD INDEX(30)

Correct HEAD : 12

HEAD Found : 18
POS : .
Correct minority voters : 2

---

stanford

desr

Committee results : [stanford-12 (93.0)][lth-18 (96.0)][idp-18 (90.0)][desr-12 (87.0)]

L'exemple ci-dessus représente l'erreur produite pour le 30e mot de la 3e phrase. La tête correcte est le mot d'index 12. La tête trouvée est le mot d'index 18. Le libellé de catégorie syntaxique du mot d'index 30 est ".". Pour ce vote erroné, deux analyseurs du comité avaient trouvé la bonne réponse, soit Stanford et DeSR. La dernière ligne présente en détail les votes du comité accompagné de la précision de chaque analyseur pour le libellé de catégorie syntaxique ".".

7.5 Plan d'expérience

Nos expériences se font en deux temps. Nous voulons d'abord identifier quelle combinaison de technique de vote (objectif O5) et de reconstruction (objectif O6) est la plus avantageuse. Par la suite, nous voulons déterminer quel comité d'analyseur performe le mieux (objectif O7).

Nous identifions deux comités à cette étape, un avec LTH et un sans. Puisque LTH est nettement supérieur aux autres analyseurs et nous le soupçonnons légèrement biaisée à cause de ses performances significativement plus élevées que les autres analyseurs, nous pensons important de faire des expérimentations pour des comités avec et sans sa présence. Ce soupçon provient surtout du fait que l'auteur de l'analyseur LTH a lui-même défini (et gagné) la campagne d'évaluation de CoNLL sur lequel l'analyseur a été évalué (Surdeanu *et al.*, 2008).

Tout ce travail est fait pour en arriver à un contexte de fusion permettant le gain de performance optimal et non biaisé. Il est important d'éliminer le plus de biais possible afin de s'assurer que notre analyseur performera aussi bien dans un contexte tout autre que CoNLL.

Nous nous attendions à ce que la combinaison de technique de vote pondérée par catégorie syntaxique avec une reconstruction par MST soit la plus avantageuse. Cette attente était basée sur le fait que ces deux techniques sont plus avancées algorithmiquement que les autres. Nous avons tout de même fait des expérimentations afin de s'assurer de ce fait.

Nous validons ce fait en choisissant simplement deux comités d'analyseurs de façon empi-

rique et en testant toutes les possibilités de combinaison de techniques de vote et de reconstruction sur nos trois corpus. Pour éviter tout biais, nous calculons toujours les pondérations sur un autre corpus que celui que nous utilisons pour la validation. Nous faisons cette expérience sur 6 jeux de données, soit sur les six paires de deux corpus possibles. Nous faisons le calcul des pondérations sur un des deux corpus et la validation de notre attente sur l'autre. Les résultats complets sur ces 6 jeux de données se retrouvent dans l'Annexe B.

Nous avons constaté que dans plus du 2/3 des cas, soit plus de 8 fois sur 12 (2 comités pour 6 jeux de données), la technique de vote pondéré par catégorie syntaxique est la plus avantageuse, ce que nous considérons comme une confirmation de notre intuition. Nous respecterons la même contrainte pour la technique de reconstruction.

Ensuite, nous voulions déterminer quel comité d'analyseurs produit les meilleurs résultats. Puisque cette notion est beaucoup moins intuitive, nous avons adopté une approche beaucoup plus rigoureuse basée sur une validation croisée.

Il y a trois étapes à la démarche générale de validation croisée. Chaque étape nécessite l'utilisation d'un corpus différent. Donc, à chaque fois qu'on applique une validation croisée, trois corpus sont nécessaires. Les deux premiers corpus sont utilisés pour des fins expérimentales. Le dernier corpus n'est aucunement utilisé durant les expériences, car nous voulons le garder sans biais pour valider les observations expérimentales.

Supposons que nous utiliserons les corpus A,B et C dans cet ordre dans l'exemple suivant. Ceci correspond à la possibilité 1 du Tableau 7.5. La première étape s'appelle l'étape **d'entraînement**. Nous calculons les pondérations des analyseurs en les faisant analyser de manière indépendante un premier corpus, le corpus A, et en évaluant les résultats.

La deuxième étape est l'étape de **sélection**. Nous configurons comboParser avec le deuxième corpus, le corpus B, en entrée. Nous avons aussi besoin des pondérations calculées à l'étape d'entraînement. Après la configuration, nous sommes prêts à exécuter comboParser. Des résultats de cette exécution, nous classifions et trions les analyseurs en fonction de leur performance et nous leur donnons un rang. Avec ces rangs, nous identifions le comité ayant le mieux performé.

Dans l'étape de **validation**, nous configurons à nouveau comboParser avec le troisième corpus, le corpus C, et nous l'exécutons. Le corpus de validation sert à confirmer que la com-

binaison gagnante à l'étape de sélection procure des résultats similaires. Nous recalculons les rangs des analyseurs en fonction de leur performance et nous validons que le meilleur analyseur identifié à la deuxième étape ait encore un rang acceptable en plus d'une performance similaire.

Étant donné que nos trois corpus ne sont pas très gros dû à l'exclusion de la section *train*, nous utilisons la validation croisée pour pouvoir utiliser le plus possible nos corpus sans engendrer de biais. Le principe est simple ; nous excluons toujours une partie des corpus disponibles que nous n'utilisons pas durant l'entraînement ou la sélection. Nous validons ensuite les conclusions que nous avons faites sur ce corpus exclu. Étant donné que ce dernier n'a pas été utilisé précédemment, nous avons la garantie que la validation se fait sans aucun biais.

Nous avons décidé d'avoir trois corpus, donc nous pouvons faire cette démarche six fois et comparer les résultats obtenus. Les six possibilités sont les suivantes :

Tableau 7.3 Approche de validation croisée pour déterminer le meilleur comité d'analyseurs

Possiblité	Entraînement	Sélection	Validation
1	A	B	C
2	A	C	B
3	B	A	C
4	B	C	A
5	C	A	B
6	C	B	A

CHAPITRE 8

ANALYSE DES RÉSULTATS ET DISCUSSION

Ce chapitre présente les travaux effectués afin d'isoler la meilleure technique de vote et la meilleure technique de reconstruction. Nous présentons aussi nos travaux qui visent à trouver la meilleure combinaison d'analyseurs afin d'optimiser le gain de performance. Finalement, nous étudions aussi l'apport de la variété algorithmique au sein d'un comité d'analyseurs.

8.1 Résultats individuels des analyseurs

Les UAS pour chaque analyseur et pour chaque corpus se retrouvent dans la table suivante, en plus de leur UAS moyen. Les performances plus détaillées comme la performance par catégorie syntaxique se retrouvent dans l'Annexe A.

Tableau 8.1 Performance individuelle des analyseurs choisis

Analyseur	UAS moyen	UAS corpus A	UAS corpus B	UAS corpus C
Stanford	75,04	75,74	74,14	75,23
Ensemble	85,04	86,03	84,82	84,27
IDP	86,97	87,78	86,32	86,80
DeSR	88,39	89,01	88,55	87,62
MSTParser	89,41	90,35	89,29	88,58
LTH	91,76	92,62	91,53	91,14

8.2 Identification des techniques de votes et reconstruction

Pour commencer, nous cherchons à isoler dans quel contexte expérimental l'analyseur combinatoire produit les résultats les plus intéressants. Pour ce faire, nous suivons la procédure expliquée à la Section 7.5. Dans nos expérimentations, nous avons choisi deux comités pour isoler le contexte. Ces deux comités sont les six analyseurs à l'exception de LTH (comité 1) et les six analyseurs à l'exception d'Ensemble (comité 2). Le comité 1 représente tous les analyseurs sauf le meilleur (LTH) et le comité 2 représente tous les analyseurs sauf le moins

bon des analyseurs basés sur les transitions (Ensemble).

Plusieurs autres combinaisons d'analyseurs ont été testées de façon empirique, mais ce sont ces deux combinaisons qui ont produit les résultats les plus représentatifs de l'ensemble et qui nous permettront de mieux expliquer nos observations. Effectivement, durant nos essais, nous avons vu que les comités de cinq analyseurs sont ceux qui produisent les résultats les plus intéressants et ces deux comités représentent les bornes inférieures et supérieures de ces comités intéressants de cinq analyseurs. Nous évaluons donc la performance de ces deux comités sur chacune des combinaisons possibles de techniques de vote et de reconstruction.

Il est à noter que nous avons vite réalisé empiriquement que le comité constitué des six analyseurs ne produit pas les meilleurs résultats. Nous pensons que ceci serait dû au fait que la présence de quatre analyseurs basés sur les transitions dans le même comité diminue grandement le gain de performance provenant de la variété algorithmique.

Nous avons trois techniques de vote et trois techniques de reconstruction, ce qui nous donne neuf possibilités de combinaison. Il est à noter que la possibilité de ne pas reconstruire l'arbre est aussi une technique de reconstruction. Par contre, nous voulons toujours produire des arbres syntaxiques valides, donc ces résultats ne nous intéressent pas, même s'ils sont évalués.

La performance de ces combinaisons de techniques est évaluée sur six jeux de données différents. Ces jeux de données sont en fait les six paires de corpus possibles avec nos trois corpus. La raison pour laquelle nous nécessitons une paire de corpus est que nous utilisons d'abord un premier corpus pour ajuster les pondérations de vote et ensuite un deuxième corpus pour valider que la combinaison de vote pondéré par catégorie syntaxique et de reconstruction par MST est bien la meilleure.

Les résultats détaillés de la performance de chaque combinaison se retrouvent dans l'Annexe B. Dans les Tableaux 8.2 et 8.2, soit une pour le premier comité et une autre pour le second, nous retrouvons la meilleure combinaison pour chaque jeu de données.

Tableau 8.2 Évaluation des performances du comboParser en fonction des stratégies de vote et de reconstruction pour le comité 1

Entraînement	Validation	Vote	Reconstruction	UAS
A	B	Pondéré UAS	MST	90,75
A	C	Pondéré UAS	MST	90,72
B	A	Pondéré POS	MST	91,90
B	C	Pondéré POS	MST	90,78
C	A	Pondéré POS	MST	91,91
C	B	Pondéré POS	MST	91,54

Tableau 8.3 Évaluation des performances du comboParser en fonction des stratégies de vote et de reconstruction pour le comité 2

Entraînement	Validation	Vote	Reconstruction	UAS
A	B	Pondéré POS	MST	91,56
A	C	Pondéré POS	MST	91,54
B	A	Pondéré POS	MST	92,96
B	C	Pondéré POS	MST	91,58
C	A	Pondéré POS	MST	92,97
C	B	Pondéré UAS	Meilleur	92,14

Nous remarquons que 9 fois sur 12, la meilleure technique de vote a été la technique de vote pondéré par catégorie syntaxique, ce qui répond à notre critère. Quant à la technique de reconstruction par MST, elle a été supérieure à la technique de reconstruction par meilleur analyseur 11 fois sur 12. Ici, nous présentons seulement les résultats de deux comités, mais dans l'ensemble de nos expérimentations empiriques, cette combinaison de technique de vote et de reconstruction a toujours montré des résultats très intéressants.

8.3 Résultats de l'analyseur combinatoire en fonction des comités

Puisque nous avons isolé la technique de vote et de reconstruction qui présente les meilleurs résultats, nous pouvons désormais faire une batterie d'expérimentations pour tester exhaustivement les performances de tous les comités de vote possibles. Tel qu'expliqué à la Section 7.5, nous faisons de la validation croisée afin de nous assurer d'avoir des résultats sans biais.

Pour chaque exécution de comboParser, soit pour chaque étape de sélection et validation,

nous devons analyser le corpus en entrée avec toutes les combinaisons d'analyseurs. Mathématiquement, ceci est représenté par la formule suivante :

Total $= \sum_{i=3}^{5} \binom{6}{i} = \frac{6!}{3!3!} + \frac{6!}{4!2!} + \frac{6!}{5!1!} = 20 + 15 + 6 = 41$ comités différents de 3 à 5 analyseurs.

Ces 41 combinaisons sont toutes testées avec la technique de votes pondérés par catégorie syntaxique et avec la reconstruction de seconde analyse (MST). Nous essayons toutes ces combinaisons 12 fois, soit une fois pour la sélection et une fois pour la validation et ce, pour les 6 phases de la validation croisée (tableau 6.2). Nous voulons trouver deux comités, soit le plus performant avec LTH et le plus performant sans LTH, comme expliqué à la Section 7.5. Il y a 41 possibilités de comités, mais quand nous excluons LTH des analyseurs potentiels, il y en a seulement 16.

Pour valider le comité trouvé durant l'étape de sélection, nous nous assurons que ce même comité est encore dans les 3 meilleurs comités durant l'étape de la validation. Donc, durant la validation, nous identifions le rang du comité trouvé durant l'étape de sélection, soit sur 41 pour le meilleur comité avec LTH, soit sur 16 pour le meilleur comité sans LTH. Nous identifions aussi le gain de performance (Δ) pour chacun des meilleurs comités.

Les résultats sont présentés dans les deux tableaux suivants, soit un pour le comité le plus performant avec LTH et le plus performant sans LTH. La colonne E représente le corpus d'entraînement, la colonne S le corpus de sélection et la colonne V le corpus de validation. Nous présentons le meilleur comité ainsi que son UAS. Ensuite, nous retrouvons la colonne représentant la performance du meilleur analyseur sur le corpus de sélection ainsi que le gain de performance (Δ). Nous donnons aussi la performance (UAS) sur le corpus de validation du comité gagnant dans la colonne "UAS validation". Finalement, nous indiquons le rang auquel s'est classé le comité parmi toutes les combinaisons sur le corpus de validation. Les résultats exhaustifs de la performance de chaque comité sont présentés dans l'Annexe C.

Tableau 8.4 Évaluation des comités les plus performants sans LTH

E	S	V	Meilleur comité	UAS combo	UAS MST	Δ	UAS validation	Rang (sur 16)
A	B	C	stanford-ensemble-idp-desr-mst	91,58	89,29	2,29	90,7	1
A	C	B	stanford-ensemble-idp-desr-mst	90,7	88,58	2,12	91,58	1
B	A	C	stanford-ensemble-idp-desr-mst	91,85	90,35	1,5	90,79	1
B	C	A	stanford-ensemble-idp-desr-mst	90,79	88,58	2,21	91,85	1
C	A	B	stanford-ensemble-idp-desr-mst	91,91	90,35	1,56	91,58	1
C	B	A	stanford-ensemble-idp-desr-mst	91,58	89,29	2,26	91,91	1
MOYENNE :						1,99		1

Tableau 8.5 Évaluation des comités les plus performants avec LTH

E	S	V	Meilleur comité	UAS combo	UAS LTH	Δ	UAS validation	Rang validation (sur 41)
A	B	C	stanford-desr-mst-lth	92,29	91,53	0,76	91,52	2
A	C	B	stanford-idp-desr-mst-lth	91,55	91,14	0,41	92,02	3
B	A	C	stanford-idp-desr-mst-lth	92,95	92,62	0,33	91,61	1
B	C	A	stanford-idp-desr-mst-lth	91,61	91,14	0,47	92,95	1
C	A	B	stanford-idp-desr-mst-lth	92,98	92,62	0,36	92,06	2
C	B	A	stanford-desr-mst-lth	92,29	91,53	0,76	92,94	2
MOYENNE :						0,52		1,83

Le comité le plus performant avec LTH est Stanford-IDP-DeSR-MSTParser-LTH. Effectivement, 4 fois sur 6, il est le comité identifié durant la phase de sélection et son rang durant la validation n'est jamais plus bas que 3e sur 41. Pour le comité sans LTH, nous avons identifié le comité de Stanford-Ensemble-IDP-DeSR-MSTParser comme le meilleur comité. Il a toujours été identifié comme le meilleur comité durant la sélection et a toujours été le meilleur comité durant la validation aussi.

8.4 Analyse des résultats

8.4.1 Performance

Trouver un analyseur avec une performance de plus de 90% (91,76% pour être plus spécifique) ne faisait pas parti de nos attentes initiales. Le meilleur gain de performance obtenu est

en moyenne de 0,50% avec LTH dans le comité. Il est très difficile d'aller chercher plus de gain de performance lorsque nous sommes déjà au-delà des 90% de performance. Nous pensons que cet analyseur est possiblement biaisé par rapport au corpus CoNLL, sans pouvoir le confirmer.

Néanmoins, il est normal que nous n'ayons pas atteint nos attentes de gain de 2 à 3%, car, comme mentionné précédemment, notre hypothèse s'appliquait pour les analyseurs de 75 à 90%, mais pas un analyseur avec au-delà de 90% de performance pratique. Par contre, dans l'optique où nous cherchons un gain de performance de 2 à 3% pour des analyseurs se situant entre 75 et 90% de UAS, nous répondons pleinement à notre objectif.

En effet, pour le meilleur comité possible excluant LTH, nous obtenons un gain moyen de 2%. Le meilleur analyseur de ce comité, MSTParser, a une performance moyenne de 89,41%. Nous pensons donc que nos techniques combinatoires confirment l'hypothèse H1.

8.4.2 Apport de la variété algorithmique des analyseurs

Si nous regardons le Tableau 8.3, nous pouvons constater que le meilleur comité est constitué de quatre analyseurs principalement et avec LTH comme meilleur analyseur, ce dernier dominant en performance individuelle. Ensuite, le comité est constitué d'un analyseur basé sur les graphes (MSTParser), un analyseur basé sur les transitions (DeSR) et un analyseur basé sur les grammaires (Stanford). Finalement, 4 fois sur 6, IDP est aussi présent dans ce comité.

Ces données semblent supporter l'hypothèse H3, qui supposait que l'utilisation d'une variété d'analyseurs basés sur des stratégies d'analyse différentes était une bonne approche au lieu de seulement se concentrer sur les analyseurs les plus performants. Pour ajouter du poids à cette observation, nous pouvons aussi observer que Stanford est présent dans les six comités alors qu'Ensemble n'y est jamais présent. Ceci est contre-intuitif puisqu'Ensemble est beaucoup plus dominant en performance (85,04%) que Stanford (75,04%).

Toutefois, ceci confirmerait que la performance individuelle des analyseurs est importante, mais la présence d'un analyseur utilisant la même stratégie d'analyse dans le comité amène un impondérable par rapport à l'importance accordée à la performance durant la sélection de l'analyseur suivant. Ainsi, l'analyseur Ensemble étant le moins bon des trois analyseurs basés sur les transitions, soit Ensemble, IDP et DeSR, mais tout de même supérieur en performance à Stanford, est souvent ignoré au dépend de ce dernier dépendamment des analyseurs sélec-

tionnés précédemment. De plus, encore selon le Tableau 8.3, Stanford a même été présent dans des comités où IDP (UAS de 86,97%) n'y figurait pas, ce qui est encore plus surprenant. Nous pouvons donc confirmer l'hypothèse H3 grâce à ces faits.

CHAPITRE 9

TRAVAUX FUTURS

Nos travaux ont amené des avancements significatifs, mais nous pensons qu'il y a moyen d'explorer encore plus en profondeur quelques pistes identifiées durant nos expérimentations. Effectivement, en produisant des statistiques détaillées, nous avons trouvé quelques pistes potentielles pour des travaux futurs traitant de l'amélioration d'analyseurs syntaxiques.

9.1 Traitement des votes minoritaires

Durant l'analyse des résultats, nous en sommes arrivés à la conclusion que dans un grand nombre de sorties erronées, des votes minoritaires corrects avaient été placés pour la bonne tête de dépendance. Par conséquent, nous pensons que des travaux peuvent être effectués pour chercher un gain de performance à ce niveau. Dans le cas de la performance maximale obtenue, soit 92,66% avec le comité Stanford-DeSR-MST-LTH, un analyseur avait trouvé la bonne réponse dans 32,11% des analyses erronées (2856/4207 mots erronés). Donc, ceci nous amène à un potentiel d'environ 95%.

Nous suggérerions une approche de renforcement plutôt que de nouvelles stratégies de fusion. Effectivement, nous ferions voter les analyseurs comme nous le faisons actuellement, mais grâce à une meilleure utilisation du contexte linguistique, nous ferions des interventions durant le vote pour ajouter du poids aux votes minoritaires dans certaines situations.

Par exemple, si nous savons qu'une réponse est toujours correcte lorsque Stanford et LTH votent tous les deux pour la même tête pour un mot de la catégorie syntaxique DET, alors nous pouvons renverser la décision du comité avec un veto. En réussissant à identifier les contextes dans lesquels nous pouvons renverser la décision du comité en fonction des votes minoritaires, nous pourrions nous approcher du potentiel de 95%.

9.2 Apprentissage automatisé

Dans nos expérimentations, nous avons établi que le vote pondéré par catégorie syntaxique, un raffinement du vote pondéré par UAS, était plus précis. Ceci nous a prouvé qu'il

y a moyen d'utiliser le contexte linguistique pour améliorer la précision de l'analyseur. De plus, comme mentionné précédemment, avec un meilleur traitement des votes minoritaires, nous pourrions nous approcher du potentiel de 95%.

Nous suggérons d'injecter dans l'algorithme des variables modificatrices qui augmentent le taux de confiance pour certaines têtes candidates en fonction de la combinaison spécifique des analyseurs ayant voté pour cette tête ainsi que de la catégorie syntaxique du mot sur lequel nous votons.

Pour ce faire, nous avons déjà mis en place un système de production de sorties binaires, représentant pour chaque vote la catégorie syntaxique du mot et les analyseurs ayant voté pour la bonne réponse. Nous pensons qu'en générant une adaptation de ces sorties avec un corpus d'entraînement, nous pourrions extrapoler la valeur des facteurs à injecter et ensuite répéter le processus avec les nouvelles variables modificatrices jusqu'à ce que nous nous stabilisions vers un gain de performance satisfaisant.

Nous pensons que l'approche d'apprentissage AdaBoost (Freund et Schapire, 1995) pourrait être très intéressante pour notre problème. Tout d'abord, nous faisons analyser un corpus d'entraînement par tous les analyseurs. Ensuite, les résultats sont séparés en jeux de données selon la catégorie syntaxique du mot analysé puisque la structure de phrase nous importe peu pour l'apprentissage des variables modificatrices.

Ensuite, grâce à l'approche AdaBoost, les nouvelles valeurs des variables modificatrices sont apprises en fonction des analyseurs ayant voté pour la bonne tête. Nous nous devons donc faire n apprentissages AdaBoost de k itérations, où n est le nombre de catégories syntaxiques et k le nombre d'analyseurs.

9.3 Tests sur d'autres corpus

Comme il a été mentionné précédemment, quelques-uns des analyseurs du comité ont été entrainés statistiquement sur la partie d'entraînement du corpus CoNLL, ce qui entraîne l'hypothèse qu'un biais statistique est peut-être présent. Pour confirmer ou infirmer ce doute présent, nous voulons évaluer les performances sur d'autres corpus. Lors de la rédaction de ce mémoire, nous sommes en train de tester les analyseurs sur le corpus SensEval-3, sous le format CoNLL.

CHAPITRE 10

CONCLUSION

Tout d'abord, nous avons été satisfaits de la performance des meilleurs analyseurs trouvés. Nous avons vu dans la littérature que la plupart des analyseurs performants se situaient entre 85 et 90%. Après nos tests pratiques, cinq d'entre eux se situaient au minimum dans cet intervalle théorique.

Par la suite, nos résultats préliminaires nous ont tout de suite indiqué qu'il était possible d'obtenir un gain de performance avec les techniques combinatoires. Ces résultats ont été obtenus empiriquement très tôt dans le cheminement et nous ont encouragé à explorer les différentes combinaisons de techniques et comités possibles.

Toutefois, dans un faible pourcentage des cas, nous avons réalisé que les résultats produits n'étaient pas des arbres bien formés. Nous avons donc testé les techniques de reconstruction d'arbre, au dépend d'une perte faible du gain de performance.

Après avoir essayé d'implémenter quelques algorithmes de reconstruction classiques, nous en sommes venus à la conclusion qu'il était possible de garder des gains de performance intéressants tout en assurant des arbres bien formés grâce à l'algorithme d'Edmonds. Nous confirmons ainsi l'hypothèse que nous pouvons produire des arbres bien formés tout en faisant une fusion d'analyseurs (hypothèse H2).

Après plusieurs tests empiriques sur différents comités, nous avons déterminé que la meilleure technique de fusion était le vote pondéré par catégorie syntaxique. Le meilleur gain de performance avec LTH présent dans le comité est d'environ 0,5%. Ce résultat semble faible, mais pour un analyseur avec une performance d'environ 92%, c'est un gain considérable. Néanmoins, comme LTH est un analyseur qui est très dominant et que nous n'avons pas pu réussir à identifier son biais d'entraînement, nous considérons aussi le meilleur gain de performance sans sa présence dans le comité, dans le cas où les performances de LTH chutent radicalement dans un contexte hors-CoNLL.

Nous obtenons un gain de performance d'environ 2% pour un comité formé d'analyseurs se situant 75 et 90% de performance, tel que supposé dans l'hypothèse H1. Le comité pro-

duisant ce résultat est constitué des cinq analyseurs dans cet intervalle, soit Stanford, DeSR, Ensemble et MSTParser. Ce gain représente une diminution du taux d'erreur d'environ 20%.

Nous avons aussi étudié l'apport de la variété algorithmique des analyseurs. Dans nos expérimentations, nous avons vu qu'un comité comprenant Stanford au lieu d'Ensemble offrait de meilleures performances que le même comité avec Ensemble. Ce qui est surprenant est que Stanford, analyseur basé sur les grammaires avec environ 75% de performance, ait été choisi au lieu de Ensemble ou IDP, les deux offrant des performances entre 85 et 90%, mais de type basé sur les transitions (comme DeSR).

Ce phénomène se retrouve souvent et nous avons étudié le nombre de fois où Stanford a été préféré à Ensemble. Ceci nous a permis de conclure qu'une variété algorithmique des analyseurs du comité apporte un gain de performance plus significatif qu'uniquement utiliser des analyseurs très performants, mais de même type d'analyse (comme supposé dans l'hypothèse H2).

En résumé, nos travaux nous ont permis de confirmer nos hypothèses initiales et de tirer des conclusions intéressantes. Notre banc d'expérimentation flexible permettra à de futurs chercheurs de creuser encore plus les techniques de fusion, d'insérer de nouveaux analyseurs performants qui apparaissent dans le domaine ou même de tester des techniques de peaufinage statistique pour la pondération des analyseurs.

RÉFÉRENCES

ABNEY, S. (1989). A computational model of human parsing. *Journal of psycholinguistic Research*, 18, 129–144.

ATTARDI, G. (2006). Experiments with a multilanguage non-projective dependency parser. *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 166–170.

BROWN, P., COCKE, J., PIETRA, S., PIETRA, V., JELINEK, F., LAFFERTY, J., MERCER, R. et ROOSSIN, P. (1990). A statistical approach to machine translation. *Computational linguistics*, 16, 79–85.

BRUNET-MANQUAT, F. (2004). Fusionner pour mieux analyser : Conception et évaluation de la plate-forme de combinaison. *Proc. TALN*. 10.

BUCHHOLZ, S. et MARSI, E. (2006). Conll-x shared task on multilingual dependency parsing. *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 149–164.

CHEVALIER, M., DANSEREAU, J., POULIN, G. et DE MONTRÉAL. GROUPE DE RECHERCHES POUR LA TRADUCTION AUTOMATIQUE, U. (1978). *TAUM-METEO : description du système*. TAUM/Université de Montréal.

COVINGTON, M. (2001). A fundamental algorithm for dependency parsing. *Proceedings of the 39th annual ACM southeast conference*. Citeseer, 95–102.

EDMONDS, J. (1968). *Optimum branchings*. National Bureau of standards.

EISNER, J. (1996). Three new probabilistic models for dependency parsing : An exploration. *Proceedings of the 16th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 340–345.

FREUND, Y. et SCHAPIRE, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. *Computational learning theory*. Springer, 23–37.

HENDERSON, J. et BRILL, E. (1999). Exploiting diversity in natural language processing : Combining parsers. *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. 187–194.

HUTCHINS, J. (1992). The first mt conference, 1952. *MT News International,*, 11–12.

HUTCHINS, J. (1999). Milestones in machine translation. *International Journal of Language and Documentation*, 20–21.

HUTCHINS, J. (2003). Alpac : the (in) famous report. *Readings in machine translation*, 131.

HUTCHINS, W. (2004). The georgetown-ibm experiment demonstrated in january 1954. *Machine Translation : From Real Users to Research*, 102–114.

JOHANSSON, R. et NUGUES, P. (2007). Lth : Semantic structure extraction using non-projective dependency trees. *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*. 227–230.

KIM, J., OHTA, T., TATEISI, Y. et TSUJII, J. (2003). Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19, i180.

KLEIN, D. et MANNING, C. (2003). Accurate unlexicalized parsing. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 423–430.

KUBLER, S., MCDONALD, R. et NIVRE, J. (2009). Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1, 1–127.

LIN, D. (1995). A dependency-based method for evaluating broad-coverage parsers. *International Joint Conference on Artificial Intelligence*. Citeseer, vol. 14, 1420–1427.

MARCUS, M., MARCINKIEWICZ, M. et SANTORINI, B. (1993). Building a large annotated corpus of english : The penn treebank. *Computational linguistics*, 19, 313–330.

MCDONALD, R., LERMAN, K. et PEREIRA, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 216–220.

NIVRE, J., HALL, J., KUBLER, S., MCDONALD, R., NILSSON, J., RIEDEL, S. et YURET, D. (2007a). The conll 2007 shared task on dependency parsing. *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Association for Computational Linguistics.

NIVRE, J., HALL, J., NILSSON, J., CHANEV, A., ERYIGIT, G., KUBLER, S., MARINOV, S. et MARSI, E. (2007b). Maltparser : A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13, 95–135.

NIVRE, J. (2007). Data Format.

RAYMOND J. MOONEY (Année inconnue). CS 388 : Natural Language Processing : Statistical Parsing.

RUHLMANN, L., OZELL, B., GAGNON, M., BOURGOIN, S. et CHARTON, E. (2010). Improving communication using 3d animation. *Knowledge-Based and Intelligent Information and Engineering Systems*, 410–419.

SAGAE, K. et LAVIE, A. (2006). Parser combination by reparsing. *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume : Short Papers on XX*. Association for Computational Linguistics, 129–132.

SURDEANU, M., JOHANSSON, R., MEYERS, A., MÀRQUEZ, L. et NIVRE, J. (2008). The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. *Proceedings of the Twelfth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 159–177.

SURDEANU, M. et MANNING, C. (2010). Ensemble models for dependency parsing : cheap and good ? *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 649–652.

TITOV, I. et HENDERSON, J. (2007). Fast and robust multilingual dependency parsing with a generative latent variable model. *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*. 947–951.

TJONG, E. (2011). About SIGNLL.

WIKIPEDIA (2011a). Google Translate.

WIKIPEDIA (2011b). Transfer-based machine translation.

WIKIPEDIA (2011c). Warren Weaver.

WOODS, W. (1968). Procedural semantics for a question-answering machine. *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 457–471.

YOENG-JIN, C. et TSENG-HUNO, L. (1965). On the shortest arborescence of a directed graph. *Saentla Sinica 4 (1965) 1396*, 1400.

ZEMAN, D. et ŽABOKRTSKÝ, Z. (2005). Improving parsing accuracy by combining diverse dependency parsers. *Proceedings of the Ninth International Workshop on Parsing Technology*. Association for Computational Linguistics, 171–178.

ANNEXE A

Performance individuelle détaillée des analyseurs

Tableau A.1 Performance en fonction de la catégorie syntaxique (POS) de l'analyseur Stanford

POS	Corpus A	Corpus B	Corpus C
NN	81	81	81
IN	74	74	72
NNP	59	55	62
DT	91	91	91
NNS	82	79	81
JJ	82	82	83
,	67	68	68
.	95	90	93
CD	74	70	73
RB	73	71	70
VBD	80	76	77
VB	96	94	92
VBZ	78	72	74
CC	61	62	62
VBN	88	86	85
TO	78	77	79
PRP	97	96	93
VBP	71	71	74
VBG	83	78	79
HYPH	5	3	2
“	77	73	71
”	84	77	74
MD	75	72	73
POS	0	1	1
:	53	47	53
PRP$	93	94	96
$	84	84	82
WDT	9	9	11
RBR	49	67	54
RP	92	93	96
JJR	76	71	66
WRB	8	5	48
WP	6	9	78
JJS	79	80	90
(	12	29	81
)	15	11	70
RBS	77	77	86
EX	100	100	100
PDT	100	75	83

Tableau A.2 Performance en fonction de la catégorie syntaxique (POS) de l'analyseur Ensemble

POS	Corpus A	Corpus B	Corpus C
NN	88	88	86
IN	81	83	81
NNP	88	86	86
DT	92	91	92
NNS	90	87	89
JJ	88	87	87
,	74	71	71
.	91	88	88
CD	88	87	89
RB	84	83	81
VBD	90	87	88
VB	98	98	98
VBZ	87	88	87
CC	79	76	74
VBN	89	90	88
TO	85	84	85
PRP	99	98	98
VBP	88	85	88
VBG	85	82	83
HYPH	18	12	13
“	75	76	71
”	81	76	76
MD	86	88	80
POS	99	98	98
:	45	68	44
PRP$	89	92	94
$	91	93	94
WDT	96	97	94
RBR	71	71	78
RP	92	93	93
JJR	82	78	75
WRB	56	63	48
WP	87	78	78
JJS	87	89	90
(	81	84	81
)	65	71	70
RBS	77	86	86
EX	100	100	100
PDT	100	75	83

Tableau A.3 Performance en fonction de la catégorie syntaxique (POS) de l'analyseur IDP

POS	Corpus A	Corpus B	Corpus C
NN	88	91	91
IN	81	79	78
NNP	88	85	87
DT	92	96	96
NNS	90	90	91
JJ	88	90	92
,	74	70	74
.	91	87	90
CD	88	91	91
RB	84	77	75
VBD	90	89	89
VB	98	97	87
VBZ	87	88	90
CC	79	77	76
VBN	89	88	87
TO	85	80	80
PRP	99	98	98
VBP	88	85	91
VBG	85	77	77
HYPH	18	97	94
“	75	70	67
”	81	81	74
MD	86	88	86
POS	99	97	98
:	45	60	48
PRP$	89	96	97
$	91	90	91
WDT	96	95	91
RBR	71	71	71
RP	92	95	92
JJR	82	79	86
WRB	56	59	46
WP	87	70	81
JJS	87	91	85
(	81	71	85
)	65	64	87
RBS	77	91	97
EX	100	100	100
PDT	100	63	79

Tableau A.4 Performance en fonction de la catégorie syntaxique (POS) de l'analyseur DeSR

POS	Corpus A	Corpus B	Corpus C
NN	93	93	92
IN	81	83	81
NNP	92	91	90
DT	97	97	96
NNS	92	90	91
JJ	94	93	94
,	74	73	72
.	89	85	85
CD	93	92	92
RB	86	85	82
VBD	89	87	87
VB	98	97	95
VBZ	86	85	86
CC	80	80	74
VBN	91	90	88
TO	84	87	84
PRP	99	99	99
VBP	84	85	83
VBG	84	83	81
HYPH	98	99	97
“	61	63	63
”	71	77	70
MD	85	84	79
POS	99	99	98
:	45	60	49
PRP$	97	97	98
$	94	95	95
WDT	96	97	94
RBR	80	81	79
RP	95	92	95
JJR	83	81	82
WRB	77	72	67
WP	89	85	79
JJS	97	93	95
(	69	80	76
)	62	78	74
RBS	77	91	94
EX	100	100	97
PDT	100	88	75

Tableau A.5 Performance en fonction de la catégorie syntaxique (POS) de l'analyseur MST-Parser

POS	Corpus A	Corpus B	Corpus C
NN	93	93	92
IN	84	82	82
NNP	91	91	88
DT	97	97	96
NNS	92	91	91
JJ	94	92	94
,	77	75	74
.	96	91	92
CD	92	92	93
RB	84	84	82
VBD	92	90	89
VB	99	98	98
VBZ	91	87	90
CC	83	80	79
VBN	92	91	88
TO	86	88	87
PRP	99	98	98
VBP	88	87	87
VBG	86	84	83
HYPH	97	98	96
“	81	76	75
”	89	83	78
MD	90	84	87
POS	98	99	98
:	53	69	44
PRP$	98	97	99
$	96	90	96
WDT	93	96	94
RBR	79	79	80
RP	94	95	98
JJR	85	82	86
WRB	66	63	64
WP	92	78	85
JJS	92	96	90
(	62	78	81
)	69	78	68
RBS	84	97	94
EX	100	100	97
PDT	88	69	96

Tableau A.6 Performance en fonction de la catégorie syntaxique (POS) de l'analyseur LTH

POS	Corpus A	Corpus B	Corpus C
NN	95	95	93
IN	87	85	95
NNP	93	91	91
DT	98	97	97
NNS	94	94	94
JJ	94	94	95
,	81	81	81
.	97	95	94
CD	92	93	93
RB	88	86	85
VBD	96	95	93
VB	99	99	99
VBZ	95	92	94
CC	87	86	85
VBN	95	94	93
TO	89	91	90
PRP	100	99	98
VBP	92	94	95
VBG	90	87	89
HYPH	98	98	97
“	81	75	77
”	86	84	78
MD	96	91	89
POS	99	99	98
:	60	74	56
PRP$	97	97	100
$	97	95	95
WDT	96	99	97
RBR	76	76	67
RP	94	94	98
JJR	90	89	88
WRB	86	78	78
WP	94	80	85
JJS	97	98	98
(	69	80	81
)	69	76	74
RBS	90	94	89
EX	100	100	100
PDT	100	81	83

ANNEXE B

Performance des différentes combinaisons de techniques

Tableau B.1 Performance des différentes combinaisons de techniques, entraînées sur le corpus A et testée sur le corpus B

Strategy	Repair	Name	UAS	BPName	BPUAS	Delta
vote	reparse	stanford-lth-idp-desr-mst	91,49	lth	91,53	-0,04
vote	best	stanford-lth-idp-desr-mst	91,53	lth	91,53	0
weighted	best	stanford-lth-idp-desr-mst	91,53	lth	91,53	0
weighted	reparse	stanford-lth-idp-desr-mst	91,53	lth	91,53	0
pos	best	stanford-lth-idp-desr-mst	91,54	lth	91,53	0,01
pos	reparse	stanford-lth-idp-desr-mst	91,56	lth	91,53	0,03
vote	best	stanford-ensemble-idp-desr-mst	90,4	mst	89,29	1,11
weighted	best	stanford-ensemble-idp-desr-mst	90,57	mst	89,29	1,28
vote	reparse	stanford-ensemble-idp-desr-mst	90,57	mst	89,29	1,28
pos	best	stanford-ensemble-idp-desr-mst	90,59	mst	89,29	1,3
pos	reparse	stanford-ensemble-idp-desr-mst	90,68	mst	89,29	1,39
weighted	reparse	stanford-ensemble-idp-desr-mst	90,75	mst	89,29	1,46

Tableau B.2 Performance des différentes combinaisons de techniques, entraînées sur le corpus A et testée sur le corpus C

Strategy	Repair	Name	UAS	BPName	BPUAS	Delta
vote	reparse	stanford-lth-idp-desr-mst	91,48	lth	91,14	0,34
weighted	best	stanford-lth-idp-desr-mst	91,53	lth	91,14	0,39
weighted	reparse	stanford-lth-idp-desr-mst	91,53	lth	91,14	0,39
vote	best	stanford-lth-idp-desr-mst	91,53	lth	91,14	0,39
pos	best	stanford-lth-idp-desr-mst	91,54	lth	91,14	0,4
pos	reparse	stanford-lth-idp-desr-mst	91,54	lth	91,14	0,4
vote	best	stanford-ensemble-idp-desr-mst	90,4	mst	88,58	1,82
vote	reparse	stanford-ensemble-idp-desr-mst	90,56	mst	88,58	1,98
weighted	best	stanford-ensemble-idp-desr-mst	90,57	mst	88,58	1,99
pos	best	stanford-ensemble-idp-desr-mst	90,59	mst	88,58	2,01
pos	reparse	stanford-ensemble-idp-desr-mst	90,69	mst	88,58	2,11
weighted	reparse	stanford-ensemble-idp-desr-mst	90,72	mst	88,58	2,14

Tableau B.3 Performance des différentes combinaisons de techniques, entraînées sur le corpus B et testée sur le corpus A

Strategy	Repair	Name	UAS	BPName	BPUAS	Delta
vote	best	stanford-lth-idp-desr-mst	92,74	lth	92,62	0,12
weighted	best	stanford-lth-idp-desr-mst	92,79	lth	92,62	0,17
vote	reparse	stanford-lth-idp-desr-mst	92,81	lth	92,62	0,19
pos	best	stanford-lth-idp-desr-mst	92,87	lth	92,62	0,25
weighted	reparse	stanford-lth-idp-desr-mst	92,88	lth	92,62	0,26
pos	reparse	stanford-lth-idp-desr-mst	92,96	lth	92,62	0,34
vote	best	stanford-ensemble-idp-desr-mst	91,63	mst	90,35	1,28
pos	best	stanford-ensemble-idp-desr-mst	91,69	mst	90,35	1,34
weighted	best	stanford-ensemble-idp-desr-mst	91,69	mst	90,35	1,34
vote	reparse	stanford-ensemble-idp-desr-mst	91,73	mst	90,35	1,38
weighted	reparse	stanford-ensemble-idp-desr-mst	91,89	mst	90,35	1,54
pos	reparse	stanford-ensemble-idp-desr-mst	91,9	mst	90,35	1,55

Tableau B.4 Performance des différentes combinaisons de techniques, entraînées sur le corpus B et testée sur le corpus C

Strategy	Repair	Name	UAS	BPName	BPUAS	Delta
vote	best	stanford-lth-idp-desr-mst	91,14	lth	91,14	0
weighted	best	stanford-lth-idp-desr-mst	91,14	lth	91,14	0
pos	best	stanford-lth-idp-desr-mst	91,55	lth	91,14	0,41
weighted	reparse	stanford-lth-idp-desr-mst	91,55	lth	91,14	0,41
vote	reparse	stanford-lth-idp-desr-mst	91,56	lth	91,14	0,42
pos	reparse	stanford-lth-idp-desr-mst	91,58	lth	91,14	0,44
vote	best	stanford-ensemble-idp-desr-mst	90,4	mst	88,58	1,82
weighted	best	stanford-ensemble-idp-desr-mst	90,57	mst	88,58	1,99
vote	reparse	stanford-ensemble-idp-desr-mst	90,59	mst	88,58	2,01
pos	best	stanford-ensemble-idp-desr-mst	90,6	mst	88,58	2,02
weighted	reparse	stanford-ensemble-idp-desr-mst	90,76	mst	88,58	2,18
pos	reparse	stanford-ensemble-idp-desr-mst	90,78	mst	88,58	2,2

Tableau B.5 Performance des différentes combinaisons de techniques, entraînées sur le corpus C et testée sur le corpus A

Strategy	Repair	Name	UAS	BPName	BPUAS	Delta
vote	best	stanford-lth-idp-desr-mst	92,74	lth	92,62	0,12
vote	reparse	stanford-lth-idp-desr-mst	92,79	lth	92,62	0,17
weighted	best	stanford-lth-idp-desr-mst	92,79	lth	92,62	0,17
weighted	reparse	stanford-lth-idp-desr-mst	92,87	lth	92,62	0,25
pos	best	stanford-lth-idp-desr-mst	92,89	lth	92,62	0,27
pos	reparse	stanford-lth-idp-desr-mst	92,97	lth	92,62	0,35
vote	best	stanford-ensemble-idp-desr-mst	91,63	mst	90,35	1,28
weighted	best	stanford-ensemble-idp-desr-mst	91,69	mst	90,35	1,34
pos	best	stanford-ensemble-idp-desr-mst	91,72	mst	90,35	1,37
vote	reparse	stanford-ensemble-idp-desr-mst	91,77	mst	90,35	1,42
weighted	reparse	stanford-ensemble-idp-desr-mst	91,9	mst	90,35	1,55
pos	reparse	stanford-ensemble-idp-desr-mst	91,91	mst	90,35	1,56

Tableau B.6 Performance des différentes combinaisons de techniques, entraînées sur le corpus C et testée sur le corpus B

Strategy	Repair	Name	UAS	BPName	BPUAS	Delta
vote	reparse	stanford-lth-idp-desr-mst	91,97	lth	91,53	0,44
vote	best	stanford-lth-idp-desr-mst	92	lth	91,53	0,47
weighted	reparse	stanford-lth-idp-desr-mst	92,02	lth	91,53	0,49
weighted	best	stanford-lth-idp-desr-mst	92,09	lth	91,53	0,56
pos	reparse	stanford-lth-idp-desr-mst	92,1	lth	91,53	0,57
weighted	none	stanford-lth-idp-desr-mst	92,11	lth	91,53	0,58
pos	best	stanford-lth-idp-desr-mst	92,14	lth	91,53	0,61
vote	best	stanford-ensemble-idp-desr-mst	91,2	mst	89,29	1,91
vote	reparse	stanford-ensemble-idp-desr-mst	91,3	mst	89,29	2,01
pos	best	stanford-ensemble-idp-desr-mst	91,42	mst	89,29	2,13
weighted	best	stanford-ensemble-idp-desr-mst	91,43	mst	89,29	2,14
pos	reparse	stanford-ensemble-idp-desr-mst	91,54	mst	89,29	2,25
weighted	reparse	stanford-ensemble-idp-desr-mst	91,54	mst	89,29	2,25

ANNEXE C

Performance des différents comités

Tableau C.1 Performance des différents comités, entraînés sur le corpus A, testés sur le corpus B

Name		UAS	Bad Trees	BPUAS	Delta
stanford-ensemble-idp-mst	4	89,29	92	89,29	0
stanford-ensemble-idp-desr	4	90,46	99	88,55	1,91
ensemble-idp-desr-mst	4	91,17	93	89,29	1,88
stanford-ensemble-desr-mst	4	91,18	94	89,29	1,89
stanford-idp-desr-mst	4	91,31	87	89,29	2,02
stanford-ensemble-idp-lth	4	91,34	59	91,53	-0,19
stanford-ensemble-desr-lth	4	91,57	64	91,53	0,04
stanford-ensemble-idp-desr-mst	5	91,58	86	89,29	2,29
ensemble-idp-mst-lth	4	91,65	57	91,53	0,12
ensemble-idp-desr-lth	4	91,66	49	91,53	0,13
stanford-ensemble-idp-mst-lth	5	91,67	62	91,53	0,14
stanford-ensemble-mst-lth	4	91,7	47	91,53	0,17
stanford-idp-mst-lth	4	91,81	50	91,53	0,28
stanford-idp-desr-lth	4	91,84	67	91,53	0,31
ensemble-idp-desr-mst-lth	5	91,92	60	91,53	0,39
ensemble-desr-mst-lth	4	91,94	49	91,53	0,41
stanford-ensemble-idp-desr-lth	5	91,96	53	91,53	0,43
idp-desr-mst-lth	4	92	63	91,53	0,47
stanford-ensemble-desr-mst-lth	5	92,02	50	91,53	0,49
stanford-idp-desr-mst-lth	5	92,08	61	91,53	0,55
stanford-desr-mst-lth	4	92,29	58	91,53	0,76

Tableau C.2 Performance des différents comités, entraînés sur le corpus A et testés sur le corpus C

Name	Size	UAS	BPName	BPUAS	Delta
stanford-ensemble-idp-desr	4	89,72	desr	87,62	2,1
stanford-ensemble-idp-mst	4	89,97	mst	88,58	1,39
stanford-ensemble-desr-mst	4	90,34	mst	88,58	1,76
stanford-idp-desr-mst	4	90,51	mst	88,58	1,93
ensemble-idp-desr-mst	4	90,54	mst	88,58	1,96
stanford-ensemble-idp-desr-mst	5	90,7	mst	88,58	2,12
stanford-ensemble-idp-lth	4	90,86	lth	91,14	-0,28
stanford-ensemble-desr-lth	4	90,93	lth	91,14	-0,21
stanford-ensemble-mst-lth	4	91,11	lth	91,14	-0,03
stanford-ensemble-idp-mst-lth	5	91,15	lth	91,14	0,01
ensemble-idp-desr-lth	4	91,17	lth	91,14	0,03
ensemble-idp-mst-lth	4	91,2	lth	91,14	0,06
stanford-idp-desr-lth	4	91,26	lth	91,14	0,12
stanford-idp-mst-lth	4	91,26	lth	91,14	0,12
ensemble-idp-desr-mst-lth	5	91,26	lth	91,14	0,12
ensemble-desr-mst-lth	4	91,29	lth	91,14	0,15
stanford-ensemble-idp-desr-lth	5	91,35	lth	91,14	0,21
stanford-ensemble-desr-mst-lth	5	91,43	lth	91,14	0,29
idp-desr-mst-lth	4	91,49	lth	91,14	0,35
stanford-desr-mst-lth	4	91,52	lth	91,14	0,38
stanford-idp-desr-mst-lth	5	91,55	lth	91,14	0,41

Tableau C.3 Performance des différents comités, entraînés sur le corpus B et testés sur le corpus A

Name	Size	UAS	BPName	BPUAS	Delta
stanford-ensemble-idp-desr	4	91,09	desr	89,01	2,08
stanford-ensemble-idp-mst	4	91,16	mst	90,35	0,81
ensemble-idp-desr-mst	4	91,59	mst	90,35	1,24
stanford-ensemble-desr-mst	4	91,73	mst	90,35	1,38
stanford-idp-desr-mst	4	91,79	mst	90,35	1,44
stanford-ensemble-idp-desr-mst	5	91,85	mst	90,35	1,5
stanford-ensemble-idp-lth	4	92,12	lth	92,62	-0,5
stanford-ensemble-desr-lth	4	92,24	lth	92,62	-0,38
ensemble-idp-desr-lth	4	92,34	lth	92,62	-0,28
stanford-ensemble-idp-desr-lth	5	92,41	lth	92,62	-0,21
ensemble-idp-desr-mst-lth	5	92,57	lth	92,62	-0,05
stanford-ensemble-idp-mst-lth	5	92,57	lth	92,62	-0,05
ensemble-idp-mst-lth	4	92,6	lth	92,62	-0,02
stanford-ensemble-mst-lth	4	92,6	lth	92,62	-0,02
stanford-idp-desr-lth	4	92,64	lth	92,62	0,02
stanford-idp-mst-lth	4	92,67	lth	92,62	0,05
stanford-ensemble-desr-mst-lth	5	92,71	lth	92,62	0,09
ensemble-desr-mst-lth	4	92,74	lth	92,62	0,12
idp-desr-mst-lth	4	92,91	lth	92,62	0,29
stanford-desr-mst-lth	4	92,94	lth	92,62	0,32
stanford-idp-desr-mst-lth	5	92,95	lth	92,62	0,33

Tableau C.4 Performance des différents comités, entraînés sur le corpus B et testés sur le corpus C

Name	Size	UAS	BPName	BPUAS	Delta
stanford-ensemble-idp-desr	4	89,81	desr	87,62	2,19
stanford-ensemble-idp-mst	4	89,89	mst	88,58	1,31
stanford-ensemble-desr-mst	4	90,34	mst	88,58	1,76
ensemble-idp-desr-mst	4	90,48	mst	88,58	1,9
stanford-idp-desr-mst	4	90,5	mst	88,58	1,92
stanford-ensemble-idp-desr-mst	5	90,79	mst	88,58	2,21
stanford-ensemble-idp-lth	4	90,9	lth	91,14	-0,24
stanford-ensemble-desr-lth	4	90,99	lth	91,14	-0,15
ensemble-idp-desr-lth	4	91,09	lth	91,14	-0,05
idp-desr-mst-lth	4	91,14	lth	91,14	0
stanford-ensemble-mst-lth	4	91,15	lth	91,14	0,01
stanford-ensemble-idp-mst-lth	5	91,18	lth	91,14	0,04
ensemble-idp-mst-lth	4	91,21	lth	91,14	0,07
stanford-idp-desr-lth	4	91,21	lth	91,14	0,07
stanford-idp-mst-lth	4	91,22	lth	91,14	0,08
ensemble-idp-desr-mst-lth	5	91,24	lth	91,14	0,1
ensemble-desr-mst-lth	4	91,27	lth	91,14	0,13
stanford-ensemble-idp-desr-lth	5	91,33	lth	91,14	0,19
stanford-ensemble-desr-mst-lth	5	91,43	lth	91,14	0,29
stanford-desr-mst-lth	4	91,48	lth	91,14	0,34
stanford-lth-idp-desr-mst	5	91,58	lth	91,14	0,44
stanford-idp-desr-mst-lth	5	91,61	lth	91,14	0,47

Tableau C.5 Performance des différents comités, entraînés sur le corpus C et testés sur le corpus A

Name	Size	UAS	BPName	BPUAS	Delta
stanford-ensemble-idp-desr	4	91,06	desr	89,01	2,05
stanford-ensemble-idp-mst	4	91,21	mst	90,35	0,86
ensemble-idp-desr-mst	4	91,55	mst	90,35	1,2
stanford-ensemble-desr-mst	4	91,64	mst	90,35	1,29
stanford-idp-desr-mst	4	91,77	mst	90,35	1,42
stanford-ensemble-idp-desr-mst	5	91,91	mst	90,35	1,56
stanford-ensemble-idp-lth	4	92,05	lth	92,62	-0,57
stanford-ensemble-desr-lth	4	92,28	lth	92,62	-0,34
ensemble-idp-desr-lth	4	92,34	lth	92,62	-0,28
stanford-ensemble-idp-desr-lth	5	92,38	lth	92,62	-0,24
ensemble-idp-desr-mst-lth	5	92,5	lth	92,62	-0,12
stanford-ensemble-idp-mst-lth	5	92,51	lth	92,62	-0,11
ensemble-idp-mst-lth	4	92,56	lth	92,62	-0,06
stanford-ensemble-mst-lth	4	92,56	lth	92,62	-0,06
stanford-idp-mst-lth	4	92,56	lth	92,62	-0,06
stanford-idp-desr-lth	4	92,65	lth	92,62	0,03
stanford-ensemble-desr-mst-lth	5	92,65	lth	92,62	0,03
ensemble-desr-mst-lth	4	92,71	lth	92,62	0,09
idp-desr-mst-lth	4	92,9	lth	92,62	0,28
stanford-desr-mst-lth	4	92,94	lth	92,62	0,32
stanford-idp-desr-mst-lth	5	92,98	lth	92,62	0,36

Tableau C.6 Performance des différents comités, entraînés sur le corpus C et testés sur le corpus B

Name	Size	UAS	BPName	BPUAS	Delta
stanford-ensemble-idp-mst	4	90,41	mst	89,29	1,12
stanford-ensemble-idp-desr	4	90,45	desr	88,55	1,9
stanford-ensemble-desr-mst	4	91,16	mst	89,29	1,87
ensemble-idp-desr-mst	4	91,18	mst	89,29	1,89
stanford-ensemble-idp-lth	4	91,26	lth	91,53	-0,27
stanford-idp-desr-mst	4	91,26	mst	89,29	1,97
stanford-ensemble-desr-lth	4	91,5	lth	91,53	-0,03
ensemble-idp-mst-lth	4	91,57	lth	91,53	0,04
stanford-ensemble-idp-desr-mst	5	91,58	mst	89,29	2,29
stanford-ensemble-idp-mst-lth	5	91,62	lth	91,53	0,09
ensemble-idp-desr-lth	4	91,64	lth	91,53	0,11
stanford-ensemble-mst-lth	4	91,72	lth	91,53	0,19
stanford-idp-mst-lth	4	91,73	lth	91,53	0,2
stanford-idp-desr-lth	4	91,77	lth	91,53	0,24
ensemble-desr-mst-lth	4	91,8	lth	91,53	0,27
stanford-ensemble-idp-desr-lth	5	91,9	lth	91,53	0,37
ensemble-idp-desr-mst-lth	5	91,91	lth	91,53	0,38
stanford-ensemble-desr-mst-lth	5	91,94	lth	91,53	0,41
idp-desr-mst-lth	4	91,95	lth	91,53	0,42
stanford-idp-desr-mst-lth	5	92,06	lth	91,53	0,53
stanford-desr-mst-lth	4	92,29	lth	91,53	0,76